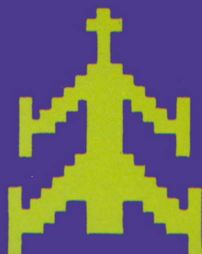
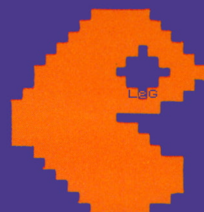
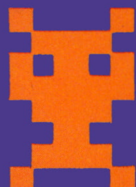
R. Rovira



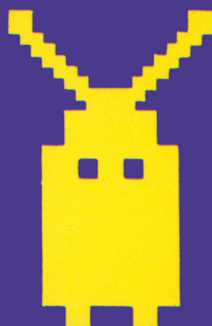


# COMO CREAR TUS JUEGOS





ZX SPECTRUM







EDITORIAL NORAY

El Corte Inglés

SOCIEDAD ESTATAL CORREOS Y TELEGRAFO
S

N.I.F. A83052407
EL CORTE INGLES S.A.
61012779 01100 1401922 220507 12:21

00797 PRODUCTOS 1 D 3,77
00/11700017

SUBTOTAL 3,77

**TOTAL 3,77**
EFECTIVO 5,00
CAMBIO EFECTIVO 1,23
. COBRADO 3,77
CT: 3
A 4/ 0,00 B 7/ 0,00 C16/ 0,00
D/ 3,77
0100 00

* GRACIAS POR SU VISITA * VIGO *
ATEN.SR/SRTA Jose Eugenio Lago

*** GRACIAS POR SU VISITA ***

# COMO CREAR TUS JUEGOS

## (ZX SPECTRUM)

# COMO CREAR TUS JUEGOS

## (ZX SPECTRUM)

**Ramón Rovira**

**EDITORIAL NORAY**

**San Gervasio de Cassolas, 79**
**08022 Barcelona**

Cualquier duda o aclaración sobre esta obra, será contestada por el departamento técnico de Editorial Noray, siempre y cuando se solicite por escrito al Apartado de correos n.º 6015, 08080 de Barcelona.


Primera edición, 1985

Depósito Legal: B. 10.556 - 1985
ISBN: 84-7486-049-0
Número de edición de E.N., 68

Printed in Spain - Impreso en España
Gráficas Instar - Industria, s/n - Hospitalet del Llobregat (Barcelona)

# PROLOGO

La idea de este libro no es la de acopiar una serie de listados de programas de juegos para que el lector pueda pasarse horas y horas tecleándolos en su Spectrum. En él se pretende enseñar los rudimentos de la programación de los juegos utilizando dos herramientas: Una es el Spectrum, uno de los microordenadores más extendidos actualmente en el mercado y la otra es el lenguaje BASIC del Spectrum. ¿Por qué el BASIC? La razón es principalmente divulgativa, puesto que es muy bajo el porcentaje de usuarios que conocen el lenguaje máquina del microprocesador Z80, mientras que la mayoría de ellos ya conocen, al menos en parte, el lenguaje BASIC.
El libro se divide en tres partes fundamentales. La primera, muy breve, está dedicada a las técnicas de programación, y en ella se explican al principio algunas de las instrrucciones más elaboradas del BASIC, dándole al lector algunas ideas de cómo usarlas en programas de juegos. La segunda parte es la más extensa y en ella se entra de lleno en la programación de los juegos que a su vez he dividido en dos grupos: Los juegos inteligentes y los juegos gráficos.
El proceso de la creación de un juego empieza desde el principio, es decir, la idea, y va siguiendo una serie de pasos hasta llegar a la culminación que es el programa. En el libro se han seguido meticulosamente todos estos pasos para la creación de cada juego. Primero se empieza con uno muy sencillo, el NIM para continuar con el OTHELLO. Después se introduce el concepto de niveles de pensamiento y se hacen las modificaciones necesarias al programa del Othello, para que utilice este nuevo concepto. Yendo un poco más lejos, llegamos al «juego del 15», que emplea un truco poco conocido que consiste en disimular un juego dentro de otro. Esto último es muy interesante y se puede aplicar a muchos otros casos e incluso permite la invención de juegos nuevos, basados en otros ya conocidos.
Después entramos en los juegos gráficos, con un único juego muy completo, llamado FANTASMAS que es una versión del popular PAC-MAN. El hecho de que haya un solo juego gráfico es debido a que el BASIC es muy lento y la mayoría de los juegos gráficos están basados en la rapidez.
La última parte del libro consiste en una serie de secciones separadas en las que se tocan distintos temas. Entre ellos están: Cómo

hacer más rápidos los programas en BASIC, cómo presentar atractivamente los programas, uso de los mandos de juego, etc.
Quizás los juegos en lenguaje máquina hubieran sido más rápidos, pero entonces sólo los entenderían unos pocos. De esta manera el que sepa programar en código máquina puede recoger la idea y realizar los programas del libro, y el que no lo conozca comprenderá perfectamente la forma en que se realiza un juego.

# ASPECTOS DE LA PROGRAMACION EN BASIC

En esta sección se comentarán algunas de las estructuras más usadas en los programas de juegos, empezando por las más sencillas, como pueden ser los contadores o el generador de números aleatorios, y terminando con las más complejas, como son los vectores y las matrices. Asimismo se incluye la explicación del funcionamiento de algunas instrucciones del BASIC del Spectrum.

## LOS CONTADORES

Un contador es una variable que se utiliza con un propósito especial, como su mismo nombre indica: Contar. El hecho de contar es muy simple y es muy utilizado en todo tipo de programas ya sea para contar puntos, fallos, posiciones en la pantalla, etc. La implementación de un contador es muy sencilla, simplemente hay que sumar a una variable el valor del paso que queremos contar. Pruebe a escribir:

```
10 LET A=1
20 PRINT A
30 LET A=A + 1
40 GOTO 20
```

La línea 10 inicializa el contador, es decir, coloca en él el valor inicial. La línea 20 actualiza el contador, esto es, le suma el paso que hemos elegido. La línea 30 hace uso del contador, en este caso, imprimiendo su valor, pero puede haber otros muchos usos:

```
10 LET A=0
20 PRINT AT 7,A; "  *"
30 LET A=A + 1
40 GOTO 20
```

En este segundo ejemplo, el contador se usa para mover un objeto en la pantalla. La línea 40 cierra el bucle del contador y hace que éste sea infinito. Es muy corriente que se desee establecer un límite

l contado, de modo que cuando se sobrepase, el programa proce la a realizar otra acción:

```
10 LET A=7
20 PRINT AT 5,A; "  *"
30 LET A=A + 2
40 IF A>30 THEN GOTO 100
20 GOTO 20
100 O T R A    A C C I O N
```

:l BASIC nos proporciona una instrucción muy potente para imple- nentar los contadores con gran facilidad. Se trata de la instrucción :OR...NEXT. Por ejemplo, el programa anterior se escribiría de la ;iguiente manera utilizando FOR...NEXT:

```
10 FOR A=7 TO 30 STEP 2
20 PRINT AT 5,A;"   *"
30 NEXT A
100 O T R A    A C C I O N
```

Los contadores se usan mucho en todos los programas aunque a veces están un poco escondidos. Para poner un ejemplo, en el programa del Othello los usaremos para contar las fichas que tiene cada jugador, para contar las jugadas, para contar las direcciones a examinar, para contar las fichas giradas, etc.

## GENERADOR DE NUMEROS ALEATORIOS

Los generadores de números aleatorios se usan para simular en un programa el efecto del azar, como por ejemplo, la tirada de un dado, la elección de un número cualquiera, el lanzamiento de una moneda, etc. El BASIC posee una instrucción que genera automáticamente un número entre 0 y 1 sin llegar a alcanzar nunva el valor 1. Se trata de la instrucción RND. Veamos cómo funciona, escriba:

```
10 PRINT RND
20 GOTO 10
```

Esto hará que se impriman en pantalla muchos números entre 0 y 1. Si nosotros queremos los números entre 0 y 10, por ejemplo, sólo tenemos que multiplicar RND por 10:

```
10 PRINT RND*10
20 GOTO 10
```

El problema está en que los números salen con decimales, pero no debemos preocuparnos, ya que disponemos de la instrucción INT que nos convierte un número decimal al entero inmediatamente inferior. Si además no queremos que salgan del 0 al 10 sino del 3 al 13 sumaremos 3 al valor modificado de RND:

```
10 PRINT 3+INT(RND*10)
20 GOTO 10
```

En general, para obtener números aleatorios entre dos números enteros N y M usaremos la siguiente fórmula:

```
10 PRINT N+INT(RND*(M-N))
```

Para poner algún ejemplo de la utilización combinada de contadores y de números aleatorios vemos estos dos pequeños programas que simulan la tirada de 5 dados y el lanzamiento de 8 monedas respectivamente:

```
10 REM TIRADA DE 5 DADOS
20 FOR I=1 TO 5
30 LET TIRADA=1+INT(RND*6)
40 PRINT TIRADA
50 NEXT I
```

```
10 REM LANZAMIENTO DE 8 MONEDAS
20 FOR I=1 TO 8
30 LET M=INT(RND*2)
40 IF M=0 THEN PRINT "HA SALIDO CARA"
50 IF M=1 THEN PRINT "HA SALIDO CRUZ"
60 NEXT I
```

La sentencia REM proviene de la palabra inglesa REMARK que significa comentario. Esta sentencia no se ejecuta y solamente sirve para dar una información, de lo que hace el programa, a la persona que ve el listado. Esta información es de gran importancia, sobre todo en los programas que no se vuelven a revisar hasta que ha pasado un período de tiempo considerable. En los programas del libro las sentencias REM se utilizan sobre todo para informar de la misión que tienen las subrutinas del programa, así como para separar unas rutinas de otras.

## LOCALIZACION Y MOVIMIENTO DE OBJETOS EN PANTALLA

El Spectrum posee una instrucción muy potente que se utiliza para saber qué carácter se encuentra en la pantalla en una posición

determinada. Se trata de la instrucción SCREEN$ que tiene como parámetros las dos coordenadas de la posición que queremos examinar y que proporciona como resultado una cadena, que contiene el carácter que se encuentra en esa posición. Su formato es:

```
SCREEN$(X,Y)
```

Para ver como funciona, teclee el siguiente programa que lo que hace es guardar la última línea de la pantalla en una cadena. Para comprobar su funcionamiento, escriba primero algún texto en la última línea de la pantalla (PRINT AT 21,0; «texto...».

```
10 LET A$="  aquí escriba 31 espacios en blanco  "
20 FOR I=0 TO 31
30 LET A$(I)=SCREEN$(21,I)
40 NEXT I
```

Si después de ejecutar este programa borra la pantalla con CLS, la última línea habrá desaparecido, pero simplemente haciendo PRINT A$ reaparecerá, aunque en otra posición. El problema que tiene SCREEN$ es que no sirve para los gráficos definidos por el usuario, o al menos, no se puede usar con facilidad sin cambiar una variable del sistema para que apunte a los gráficos definidos y aun así presenta algunos problemas, ya que después no aparecen los caracteres normales. Para solventar este problema disponemos de la instrucción ATTR, que se usa con la misma sintaxis, pero que proporciona un número compuesto por los atributos de la posición: Color del papel, color de la tinta, brillo, inverso y parpadeo.
Para mover un carácter en la pantalla simplemente hay que imprimirlo en una posición, cuidando antes de borrarlo de la posición anterior en que se había impreso. Como ejemplo sencillo puede servir el segundo programa de este capítulo que se refería a los contadores.
Otra instrucción muy interesante es la instrucción INKEY$ que es en realidad una función que proporciona el carácter del teclado que se está pulsando en ese momento. Veamos un ejemplo en que la instrucción INKEY$ se usa para controlar las teclas del cursor, con objeto de hacer un dibujo en la pantalla:

```
10 LET X=128
20 LET Y=88
30 PLOT X,Y
40 IF INKEY$="" THEN GOTO 40
50 LET A$=INKEY$
60 IF A$="5" THEN LET X=X-1
70 IF A$="6" THEN LET Y=Y+1
```

```
80 IF A$="7" THEN LET Y=Y-1
90 IF A$="8" THEN LET X=X+1
100 GOTO 30
```

¿Le parece extraña la línea 40? Esto se hace porque la instrucción INKEY$ no se espera a que usted pulse una tecla, sino que el programa «pasa de largo» en caso de que no lo haga. Otra utilidad de INKEY$ es que con ella se pueden pedir datos que consten de una sola letra sin que sea necesario pulsar ENTER una vez se haya terminado.

## CONJUNTOS

Como nos enseñaron cuando íbamos a la escuela: «un conjunto es una reunión de elementos». En el ordenador también podemos tener conjuntos de variables. La ventaja de disponer de ellos, es que el nombre de sus elementos consta de la misma letra, variando sólo los números que le siguen. Este tipo de conjuntos, también son llamados vectores o matrices. Llamaremos dimensión de una matriz, al número de cifras que son necesarios para determinar uno de sus elementos. Dado que las matrices pueden ocupar mucha memoria, es necesario avisar al ordenador de que vamos a utilizarlas. Para ello se usa la instrucción DIM. Por ejemplo la sentencia: DIM a(7) creará una matriz de 1 dimensión (las matrices de una dimensión también se llaman vectores) con 7 elementos, cuyos nombres son:

```
a(1) a(2) a(3) a(4) a(5) a(6) a(7)
```

En cambio, la instrucción DIM C(4,3) creará una matriz de dos dimensiones con 12 (3×4) elementos:

```
C(1,1) C(1,2) C(1,3)
C(2,1) C(2,2) C(2,3)
C(3,1) C(3,2) C(3,3)
C(4,1) C(4,2) C(4,3)
```

Una de las grandes ventajas de la utilización de matrices reside precisamente en el hecho de que cada elemento se designa con el mismo nombre, variando sólo los números encerrados entre paréntesis. Gracias a ello se puede evitar un trabajo muy penoso en la programación de la entrada de datos de un número elevado de variables, utilizando contadores. Veamos, como ejemplo, un programa que efectúa la entrada de datos por el teclado de una matriz de 20×30 (¡¡tiene 600 elementos!!):

```
10 DIM A(20,30)
20 FOR I=1 TO 20
30 FOR J=1 TO 30
40 INPUT A(I,J)
50 PRINT "A";I;J;"= ";A(I,J)
60 NEXT J
70 NEXT I
```

Si esto mismo se hubiera hecho usando variables sencillas se habrían tenido que teclear 600 líneas de programa con la instrucción INPUT. Las matrices se usan para muchas cosas, una de las más importantes para nosotros será la de representar tableros de juego. Pongamos por ejemplo un tablero de ajedrez. Está claro que podrá ser representado por una matriz de 8×8. Para que cada casilla represente una ficha debemos determinar una norma. Por ejemplo haremos que las fichas blancas sean números cuya primera cifra es 1, y para las negras, su primera cifra será 2, utilizando además el siguiente convenio.

1 peón
2 alfil
3 caballo
4 torre
5 dama
6 rey
0 casilla vacía

Así por ejemplo la pieza 23 es un caballo negro.
En este caso, no hay necesidad de pedir los valores de la matriz por teclado, ya que como queremos representar un tablero al principio de la partida, ya sabemos la posición que ocupan las piezas. En estos casos es muy adecuada la instrucción READ, que siempre está relacionada con las instrucciones DATA y RESTORE.
El formato de la instrucción READ es el siguiente:

```
READ variable            o también
READ var1, var2,..........varN
```

Lo que hace esta instrucción es buscar una determinada línea DATA en la que se encuentran una o varias constantes, leerla y colocar el valor leído en la variable de la instrucción READ. Este proceso es secuencial, con ello quiero decir, que cada instrucción READ lee un dato de una línea DATA, luego, el siguiente READ lee el siguiente dato y así sucesivamente. Cuando se termina una línea DATA se continúa con la siguiente, y si no hay suficientes se produce un error. Para poder cambiar esta secuencia tenemos la instrucción RESTORE, cuyo formato es:

```
RESTORE número de línea
```

Esto hace que la siguiente instrucción READ lea de una línea DATA concreta. Si se suprime RESTORE se empieza por el primer DATA que haya en el programa. Con todo esto, el programa para llenar una matriz que haga la función de tablero de ajedrez, quedaría de la siguiente manera:

```
10 DIM T(8,8)
20 DATA 24,23,22,25,26,22,23,24
30 DATA 21,21,21,21,21,21,21,21
40 DATA 0,0,0,0,0,0,0,0
50 DADA 0,0,0,0,0,0,0,0
60 DATA 0,0,0,0,0,0,0,0
70 DATA 0,0,0,0,0,0,0,0
80 DATA 11,11,11,11,11,11,11,11
90 DATA 14,13,12,15,16,12,13,14
100 FOR I=1 TO 8
110 FOR J=1 TO 8
120 READ T(I,J)
130 NEXT J
140 NEXT I
```

Para terminar con este capítulo introductorio, me gustaría dar algunos consejos para una programación cómoda y sin problemas. De entrada cuando vaya a realizar un programa, lo primero que debe hacer es alejarse lo más posible del ordenador, coger papel y lápiz, y retirarse a un lugar tranquilo donde pueda pensar con claridad. Al principio piense sólo en la idea principal del programa y no se preocupe en absoluto de la presentación ni de las otras pequeñas cosas que se le puedan ocurrir. Una vez hecho esto, desarrolle su idea sobre el papel. Para esto existen muchos métodos, uno de los cuales se explica en el capítulo siguiente y consiste en dibujar un diagrama y en escribir el programa en pseudo-lenguaje antes de codificarlo en BASIC. Desentiéndase de problemas secundarios de la idea principal y considere que ya están resueltos, luego ya los revisará uno por uno. Si no lo hace así, se encontrará con que cada uno de estos problemas secundarios le distrae de la idea principal, de manera que cuando ha resuelto uno ya ha perdido el hilo del «núcleo» del programa. Con esto termina este capítulo y entramos de lleno en la programación de los juegos.

# JUEGOS INTELIGENTES

Cuando se oye hablar de juegos inteligentes, uno piensa siempre en el ajedrez, que es un programa realmente complicado por la gran cantidad de combinaciones que existen en el juego. Un juego inteligente es, en general, aquel en el que el ordenador compite contra el usuario, y recibe el nombre de inteligente porque el ordenador es capaz de «pensar» su jugada e incluso de derrotar al contrincante. En definitiva, son todos aquellos juegos en los que el ordenador imita o intenta imitar la habilidad mental que tienen las personas con el objeto de ganar una partida.
Aunque no lo parezca, el factor tiempo es de gran importancia en este tipo de juegos, ya que son interactivos y hay que buscar un método de modo que no sea necesario repasar todas las posibilidades para ejecutar una tirada. De todos modos el tiempo de respuesta hay que ahorrarlo en el momento de diseñar el sistema de razonamiento del programa, y no hay que reparar en él, cuando el desperdicio de tiempo se da en beneficio de una mejor claridad del programa, es decir, de la estructuración.
Antes de empezar con un pequeño programa de ejemplo es interesante remarcar una serie de aspectos comunes en la programación de este tipo de juegos:

## CONOCIMIENTO DEL JUEGO

Lo primero de todo, y lo más importante es que para programar un juego de esta clase, hay que saberlo jugar si no muy bien, por lo menos con una gran seguridad, si usted no sabe lo suficiente puede servirle de gran ayuda el consejo de una persona experta. En definitiva se trata de conocer las jugadas más corrientes, las posiciones más ventajosas y también las posibles respuestas del contrario en situaciones especiales, así como el desarrollo y las fases de que se compone una partida.

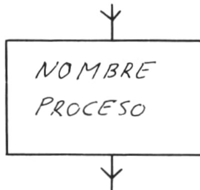
## DISEÑO DE LA ESTRUCTURA. ORGANIGRAMA

El paso siguiente consiste en diseñar lo que se llama organigrama que no es más que un gráfico de la estructura del juego. Este paso

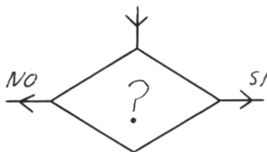
es muy importante y habrá que prestarle especial interés a la parte en la que el ordenador se «piensa» su jugada, pues de ello depende la calidad del juego. En este punto es donde hay que calibrar por un lado el tiempo de respuesta del ordenador, y por el otro lo buena que es su jugada. Por lo tanto, dentro de este apartado, o en el siguiente habrá que especificar claramente el método seguido por el ordenador para elegir su mejor jugada, hay muchos métodos generales pero aquí hablaré de dos, uno muy simple que consiste en la generación aleatoria de jugadas que se pueden pasar por varios filtros para conseguir la mejor, y el otro que se llama algoritmo (1) alfa-beta, que es capaz de tener en cuenta las posibles respuestas del adversario.
El diseño de esta estructura es gráfico, ya que de este modo permite ver con claridad la línea general del programa y para ello se emplean unos símbolos más o menos estandarizados que dan una indicación del tipo de proceso que se sigue. Para que no haya lugar a dudas voy a explicar ahora mismo en qué consisten estos símbolos:



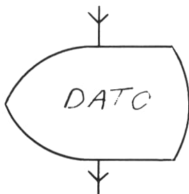

Este símbolo nos indica que hay un proceso de entrada de datos. En su interior se escribe una palabra que dé una idea del dato o los datos que se van a introducir en ese momento.



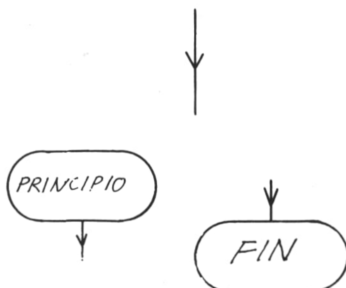

El rectángulo indica un proceso que puede ser más o menos complicado, en su interior se escribe el nombre del proceso que a su vez puede constar de un diagrama tan largo como el del propio programa, pero de esto ya hablaremos más adelante.




El rombo nos indica que en ese punto del programa hay que tomar una decisión, esto se consigue mediante la comprobación de una condición. Del rombo parten dos líneas que van a parar a dos sitios distintos, una al proceso que debe ejecutarse si la condición se verifica, y la otra al proceso que debe ejecutarse en caso contrario.




Este símbolo nos indica que en este punto del programa tiene lugar una salida de datos. El sistema de identificación es el mismo que para el de la entrada de datos.




La línea nos indica la dirección que sigue el proceso, y queda claramente especificada por el sentido de la flecha.

Por último este símbolo se usa para indicar el principio o el fin del programa. Es útil realmente en los casos de programas que tengan más de un punto de entrada o de salida.

## AFINACION O DISEÑO ARRIBA-ABAJO

Una vez realizado el diagrama del programa se sigue un proceso de afinación. Esto consiste en especificar claramente en qué consiste cada uno de los procesos, es decir de los símbolos que se usan en el diagrama, lo cual se puede hacer a su vez usando estos mismos gráficos ya que con ellos se pueden especificar todos los casos posibles que se puedan dar. Esto se hace hasta que el proceso esté tan bien definido que ya se pueda escribir un pseudo-programa, es decir, un programa en un lenguaje corriente que luego ya sólo tendremos que traducir al BASIC. Todo esto se verá con más claridad al desarrollar los ejemplos.

## ESCRITURA DEL PROGRAMA

Una vez se tiene escrito el programa en este lenguaje comprensible que hemos llamado pseudo-lenguaje, se escribe el programa simplemente traduciendo al BASIC. Por ejemplo, una instrucción en pseudo-lenguaje, podría ser:

```
PREGUNTA QUIEN EMPIEZA
```

y su traducción al BASIC del Spectrum sería algo así:

```
30 INPUT "Quieres empezar? (S/N)";a$
40 IF a$<>"S" AND a$<>"N" THE GOTO 30
50 IF a$="S" THEN GOTO ...
```

Está claro que será más fácil modificar un programa, si previamente lo tenemos escrito en pseudo-lenguaje, ya que al traducirlo a BASIC nos encontramos con que una simple instrucción como puede ser una pregunta no se traduce directamente en una sentencia INPUT, sino que hay que hacer unos filtrados de la respuesta, escoger la variable, etc.

Todos estos pasos van encaminados a conseguir que el programa tenga los menos errores posibles y a que en caso de que los haya sea fácil de depurar y de corregir, incluso si ha pasado mucho tiempo desde que se escribió.

# UN EJEMPLO SENCILLO: NIM

Para empezar con un ejemplo sencillo lo haremos con un juego llamado NIM, también conocido por «el juego de los palillos». Durante el desarrollo del programa seguiremos todos los pasos explicados. Una vez terminado, continuaremos con un juego más complejo que desarrollaremos más a fondo. Tanto en el uno como en el otro, no se presta excesiva atención a la presentación por dos razones: La primera porque ya se dedica un capítulo especialmente a este tema y la segunda porque puede distraernos del objetivo principal, que es, en este caso, que el ordenador juegue y... ¡Que juegue bien!

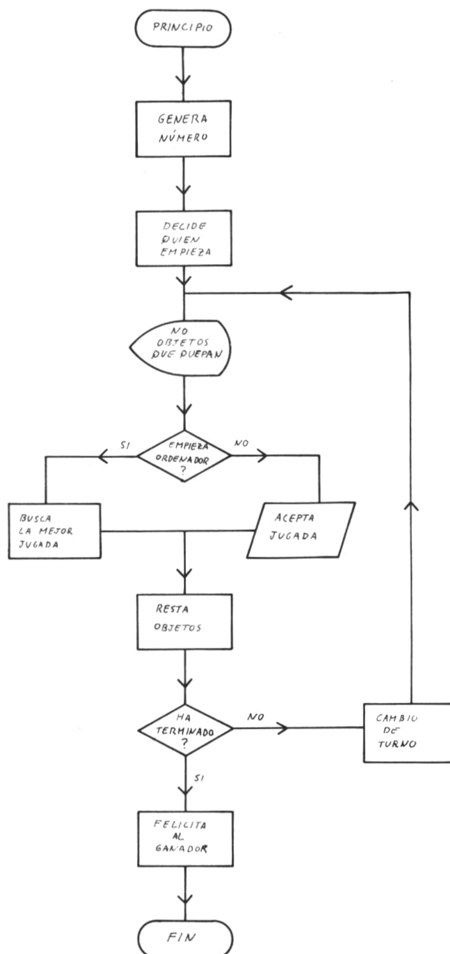
## DESCRIPCION DEL JUEGO

Se tienen una serie de objetos, por ejemplo, un número indeterminado entre 10 y 50. De ellos hay que ir retirando por turnos una cantidad no superior a 5 cada vez y el jugador que retira el último pierde. Con la particularidad de que es obligatorio retirar al menos 1 objeto. Si pensamos detenidamente en el juego observaremos que tiene la particularidad de que quien empieza, si juega bien, puede ganar seguro. ¿Por qué? Pues bien, pensemos: En primer lugar si en la última jugada, le dejamos al contrario menos de 7 objetos, por ejemplo 4, él retirará 3, y nosotros nos tendremos que quedar con el último, pero si en cambio le dejamos exactamente 7, él como mucho puede retirar 5, con lo cual quedarán 2, entonces retiraremos 1 y él se quedará con el último. En caso de que le dejemos más de 7, por ejemplo 10, entonces él puede retirar 3 y dejarnos a nosotros con 7, con lo cual perderíamos. Por tanto, es cuestión de intentar dejar al contrario con 7 objetos, así tenemos la partida ganada. Pero... ¿Cómo podemos conseguir dejarle con 7? Pues bien, habiéndole dejado previamente con 13, ya que en este caso, como que él tiene que retirar un número de objetos comprendido entre 1 y 5, nosotros siempre encontraremos un número menor que 5 que lo deje a él con 7 objetos. Si continuamos razonando del mismo modo observaremos que el siguiente número interesante de objetos es 19, el siguiente 25, etc. Es decir, que los números siguen la siguiente tónica: 1, 1+6, 1+6+6, 1+6+6+6, etc. Si suponemos que el contrario es un jugador perfecto, sólo podremos perder en

dos casos: Cuando empiece él y la cantidad de objetos no sea una de las mencionadas o bien cuando ocurra que aunque empecemos nosotros, tengamos la mala suerte de que el número de objetos a retirar sea del tipo 1+6×X cualquiera que sea X. Por lo tanto, es muy fácil hacer un programa que gane siempre. Para evitar esto haremos que el ordenador escoja aleatoriamente quién empieza.

## DISEÑO DEL GRAFICO

Dado que el programa es tan sencillo, el gráfico es también sencillo, hasta incluso se podría pensar que no es ni siquiera necesario, pero para seguir el método paso a paso lo dibujaremos:



## AFINACION

El primer proceso consiste en generar un número, que deberá estar entre 10 y 50, este número lo colocaremos en la variable N. Para decidir aleatoriamente quién empezará a jugar utilizaremos también la generación de un número que sólo tenga dos posibilidades, el −1 o el 1 y lo pondremos en la variable A.
Para decidir quién empieza sólo habrá que ver el valor de la variable A. Si vale −1 empezará el ordenador, y además se presentará en pantalla quién juega primero.
La búsqueda de la mejor jugada no es nada complicada teniendo en cuenta todo lo que sabemos del juego. Consiste en buscar un número, que restado de N nos dé un número del tipo 1+6×X. Si alguno es aficionado a las matemáticas observará que el resto de dividir N−1 por 6 es el número buscado. Para los que no lo sean no es muy difícil buscar todas las posibilidades, puesto que sólo podemos escoger entre 5 números para restar, y una vez restados de N debe ser igual a alguno de éstos: 1, 7, 13, 19, 25, 31, 37, 42 o 48, pero evidentemente esta segunda opción es mucho más lenta e inadecuada. ¡Imagínese que pudiéramos retirar hasta 20 objetos de golpe y que el número de objetos fuera de 1000! En cualquier caso también la explicaremos porque servirá para comparar.
Para la entrada del número de piezas retirado por el jugador sólo hay que mirar que se encuentre entre los límites permitidos, es decir, que sea mayor que 1 y menor o igual que 5.
El hecho de restar los objetos es igual para uno que para otro, pero habrá que decir los que quedan.
La comprobación de si se ha terminado el juego simplemente consiste en ver si queda un solo objeto, en caso contrario habrá que hacer dos cosas: La primera cambiar de turno, que no es más que cambiar el valor de A, y la segunda es ir al lugar del programa en que se decide quién tira.
Si sólo queda un objeto es que alguien ha ganado, y para saber quién ha sido sólo tenemos que mirar el valor de la variable A. Una vez ya lo tenemos todo especificado, escribimos el programa en pseudo-lenguaje:

```
GENERA UN NUMERO ENTRE 10 Y 50 (N)
GENERA UN NUMERO QUE SEA 1 O −1(A)
*SI A ES −1 HAZ LO SIGUIENTE:
BUSCA EL RESTO DE DIVIDIR N−1 POR 6 (J)
SI TE DA CERO INVENTA CUALQUIER NUMERO ENTRE 1 Y 5
ESCRIBE LA CANTIDAD QUE RETIRAS (J)
SI A ES 1 HAZ LO SIGUIENTE:
PIDE EL NUMERO DE OBJETOS A RETIRAR (J)
COMPRUEBA QUE ESTE ENTRE 1 Y 5
ESCRIBE DICHO NUMERO (J)
```

RESTA LA TIRADA DE N (N=N−J)
SI N NO ES 1 HAZ LO SIGUIENTE:
CAMBIA EL TURNO
SALTA A *
FELICITA AL ORDENADOR SI A ES 0
FELICITA AL JUGADOR SI A ES 1
TERMINA

Una vez hecho todo esto la confección del programa consiste simplemente en una traducción:

```
  10 REM ***********************
  20 REM GENERA UN NUMERO
  30 REM ***********************
  40 REM
  50 LET N=INT (40*RND)+10
  52 REM
  53 REM ***********************
  54 REM QUIEN EMPIEZA
  56 REM ***********************
  57 REM
  58 LET C=RND
  60 IF RND<0.5 THEN LET A=-1
  65 IF RND>0.5 THEN LET A=1
  70 PRINT "HAY ";N;" OBJETOS"
  80 IF A=-1 THEN PRINT "EMPIEZO
 YO"
  90 IF A=1 THEN PRINT "EMPIEZAS
 TU"
  99 REM
 100 REM ***********************
 105 REM DECIDE QUIEN JUEGA
 110 REM ***********************
 115 REM
 117 IF A=-1 THEN GO SUB 800: RE
M ORDENADOR
 120 IF A=1 THEN GO SUB 900: REM
 JUGADOR
 126 REM
 130 REM ***********************
 140 REM RESTA NUMERO
 150 REM ***********************
 160 REM
 170 LET N=N-J
 175 PRINT AT 10,15;"          "
 180 PRINT AT 10,15;"QUEDAN ";N
 200 REM
 210 REM ***********************
 215 REM MIRA SI HA TERMINADO
 220 REM ***********************
 230 REM
 240 IF N=1 THEN GO TO 300
 250 LET A=A*-1
 270 GO TO 100
 299 REM
 300 REM ***********************
 310 REM FELICITACIONES
 320 REM ***********************
 330 REM
```

```
 340 IF A=1 THEN PRINT "MUY BIEN
, HAS GANADO"
 350 IF A=-1 THEN PRINT "LO SIEN
TO, HE GANADO"
 370 STOP
 800 REM
 802 REM ************************
 805 REM TIRADA DEL ORDENADOR
 810 REM ************************
 811 REM
 815 LET COCIENTE=INT ((N-1)/6)
 820 LET RESTO=N-1-COCIENTE*6
 825 IF RESTO=0 THEN LET RESTO=I
NT (RND*5)+1
 830 LET J=RESTO
 835 BEEP .1,25: PRINT AT 2,15;"
RETIRO ";J
 840 RETURN
 900 REM
 910 REM ************************
 920 REM TIRADA DEL JUGADOR
 940 REM ************************
 950 REM
 960 INPUT "CUANTOS RETIRAS ";J
 970 IF J>5 OR J<1 OR N-J<1 THEN
GO TO 960
 980 RETURN
```

## COMENTARIOS SOBRE EL PROGRAMA

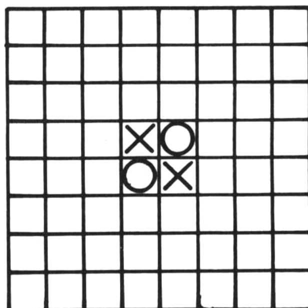
En la línea 50 se genera un número entre 10 y 50. Esto se consigue con la instrucción RND que genera un número entre 0 y 1, si lo multiplicamos por 40 nos queda entre 0 y 40, al sumarle 10 el número estará entre 10 y 50 sin alcanzar nunca este último valor, ya que RND no alcanza tampoco el valor 1. Para ver quién empieza, se pone en A el valor 1 o −1 utilizando la misma instrucción RND (línea 58). Se han utilizado dos subrutinas, una para la jugada del ordenador, y otra para la del jugador (líneas 800 y 900 respectivamente) al jugador, se le impide que diga un número que no está en el rango 1-5 así como que pierda retirando la última ficha sin querer (línea 970). En la jugada del ordenador se calcula el resto de dividir N-1 por 6 (líneas 815 y 820) si este resto da 0 quiere decir que estamos ante un número de la forma 1+6×X con lo cual el ordenador tiene las de perder, en este caso, dada la imposibilidad de conseguir la mejor jugada, el ordenador retira cualquier número entre 1 y 5 (línea 825).
En el programa hay gran cantidad de líneas REM que son comentarios que no se ejecutan. Estos comentarios van muy bien cuando se quiere retocar el programa después de que haya transcurrido un tiempo desde que se escribió. Consultar el capítulo dedicado a programación.

# EL OTHELLO: UN JUEGO DE VERDAD

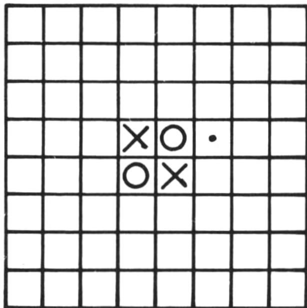
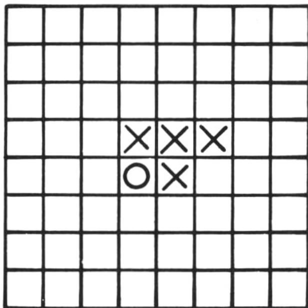
El Othello es un juego de origen oriental que se ha puesto de moda en los últimos años. De él haremos dos programas, que se diferenciarán únicamente en la parte más importante: el modo en que el ordenador piensa su jugada. En el primero, el ordenador sólo se fijará en la mejor posición a la que puede tirar, mientras que en el segundo haremos uso del algoritmo alfa-beta, que le permitirá tener una visión «de futuro» cuando realice su jugada. Para ello explicaremos a fondo en qué consiste este algoritmo antes de empezar con el segundo programa.
Por el momento vamos a por la descripción del juego, que es el requisito indispensable para poderlo programar.

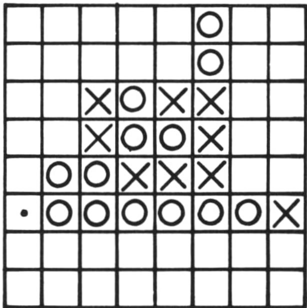
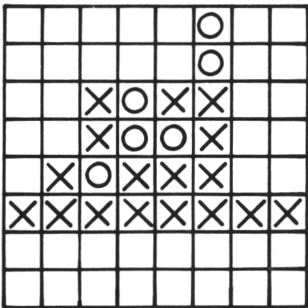
## DESCRIPCION DEL JUEGO

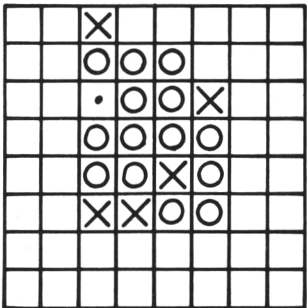
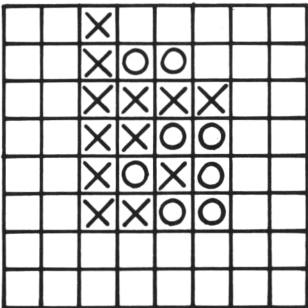
El Othello se juega en un tablero de 8×8 cuadrados sin distinción alguna entre ellos, es decir, que todos son del mismo color. Hay 64 fichas iguales pero con una particularidad, por una cara son blancas y por la otra son negras. Cada jugador escoge un color al principio de la partida, y el objetivo final es tener más fichas de su color al final de ésta. Para ello, en cada jugada se coloca una ficha en el tablero, de modo que aprisione a una o varias fichas del contrario entre dos de las propias y en cualquier dirección. A las fichas que hayan sido aprisionadas, se les cambia la cara, con lo cual pasan a tener el color del jugador.
La partida empieza con 4 fichas situadas en el tablero de la siguiente forma:



Nosotros en lugar de emplear colores blanco y negro emplearemos círculos y equis. A partir de esta disposición, el jugador que empiece ya puede tirar. Veamos algunos ejemplos de tiradas para salir de dudas. En los tres ejemplos que siguen tenemos a la izquierda una disposición inicial, en donde se señala con un punto, el lugar donde se tirará la ficha (siempre «X») y a la derecha está situado el mismo tablero pero con la ficha ya jugada y habiendo ya realizado los cambios de cara necesarios:

Otra regla importante del juego es que si en el turno de un jugador, éste no puede colocar su ficha en alguna posición en que gane fichas del contrario, el jugador pierde su turno. Es decir, que se está obligado a ganar fichas, si no no se puede tirar.
Con esto creo que quedan claras las reglas del juego, ahora vamos a ver cómo podemos desarrollar una estrategia que juegue lo suficientemente bien.
Si pensamos en una ficha cualquiera situada en el tablero, observaremos que durante el desarrollo de la partida, puede cambiar muchas veces de bando, excepto en el caso de que esa ficha esté situada en una esquina ya que una vez allí ya no se la puede aprisionar entre dos fichas enemigas. Por tanto, es muy interesante conseguir las esquinas. También se ve claramente que las fichas colocadas en los laterales sólo pueden ser aprisionadas en una dirección, y además sirven de base para girar las fichas del adversario.
Hay que tener muy en cuenta que las posiciones adyacentes a las esquinas son muy malas, incluyendo las dos laterales, pues pueden dar oportunidad al adversario de conseguir las esquinas. Por la misma razón, son bastante buenas las posiciones que están a dos lugares de una esquina porque hacen que el adversario tenga la posibilidad de tirar en una posición adyacente a la esquina lo cual como ya hemos explicado, sería malo para él. Esto aún tiene más importancia hacia el final de la partida, ya que quedan menos posiciones libres para tirar. En el gráfico siguiente aparecen las posiciones del tablero con un número situado encima. Estos números van del 8 al 1 y establecen una puntuación para cada casilla basándose en los criterios mencionados (a mayor puntuación, mejor es la casilla).

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 8 | 2 | 7 | 5 | 5 | 7 | 2 | 8 |
| 2 | 1 | 3 | 3 | 3 | 3 | 1 | 2 |
| 7 | 3 | 6 | 4 | 4 | 6 | 3 | 7 |
| 5 | 3 | 4 | | | 4 | 3 | 5 |
| 5 | 3 | 4 | | | 4 | 3 | 5 |
| 7 | 3 | 6 | 4 | 4 | 6 | 3 | 7 |
| 2 | 1 | 3 | 3 | 3 | 3 | 1 | 2 |
| 8 | 2 | 7 | 7 | 5 | 7 | 2 | 8 |

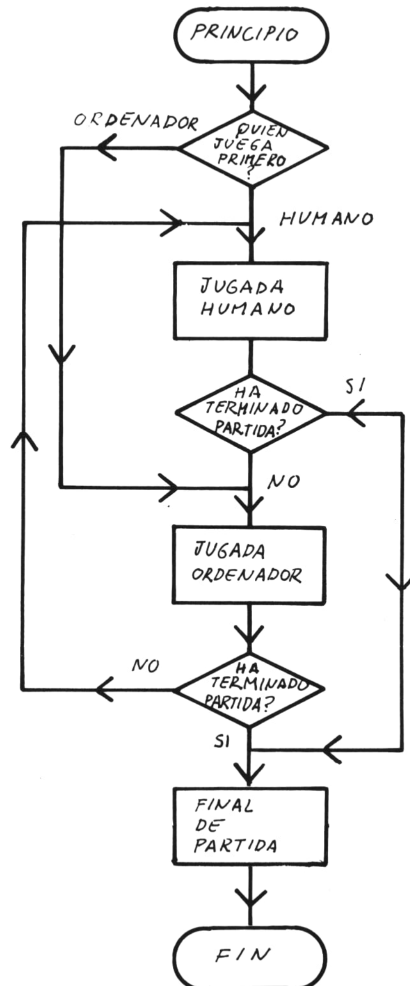
El número de fichas que se consiguen en una jugada no es un factor excesivamente importante al principio de la partida, pero va adquiriendo importancia hacia el final. Otro factor importante es el número de fichas que nos puede girar el contrario en su respuesta pero de esto ya hablaremos más adelante.
Vamos a empezar a trabajar en el primer programa de los dos que haremos para el Othello. En este primer programa, que será más complicado que el del juego del NIM, sólo tendremos en cuenta la

posición que sea más ventajosa. Es decir, que para seleccionar la mejor jugada del ordenador haremos lo siguiente:

1. Le introduciremos al ordenador todas las casillas del tablero, ordenadas de mayor a menor puntuación según el gráfico.

2. Examinaremos la casilla que tenga mayor puntuación, y si se puede tirar en ella lo haremos. En caso de que no se pueda tirar, ya sea porque esté ocupada o porque no se pueda girar ninguna ficha del contrario entonces examinaremos la siguiente que tenga mejor puntuación.

Para ello empezaremos por confeccionar un organigrama del juego, primero muy general, para irlo afinando poco a poco. Un gráfico general del desarrollo del juego podría ser algo como esto:



Este primer diagrama es muy general y nos va a servir para los dos programas que haremos, e incluso hubiera servido para el programa del NIM. Ahora vamos a especificar qué es lo que hace cada proceso para poder desglosar cada uno. Para ello se han incluido unos números en la parte inferior izquierda de cada proceso, que servirán de referencia para su posterior desglose.

## INICIALIZACION

Consiste en todo lo que se debe hacer antes de empezar a jugar la partida:

— Dibujar el tablero.
— Introducir en una variable el orden de las casillas según su puntuación.
— Visualizar en pantalla la posición inicial.
— Poner el contador de fichas «X» en 2
— Poner el contador de fichas «O» en 2
— Visualizar los contadores.
— Visualizar unas pequeñas instrucciones.

Este proceso no necesita de ningún gráfico que lo represente, pues ya es lo suficientemente claro.

## PARA DECIDIR QUIEN EMPIEZA

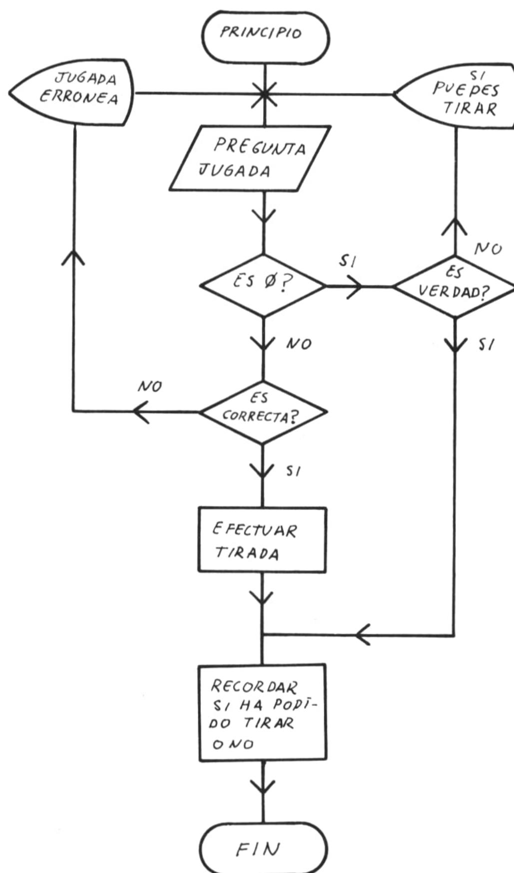
Para decidir cuál de los dos jugadores empieza, se puede hacer aleatoriamente como en el juego del NIM, o bien darle la oportunidad de decidirlo al propio jugador. En este juego optaremos por esta segunda posibilidad, por lo tanto sólo habrá que hacer una pregunta y evitar que se den respuestas incoherentes.

## JUGADA DEL JUGADOR

Este proceso puede ser muy elaborado dependiendo de los controles que se quieran imponer. Como datos de entrada de la jugada sólo están las dos coordenadas del tablero en que se quiere situar la ficha. Seguidamente hay que controlar si esa tirada es posible. Para hacer esto se deben verificar tres cosas: Primero que las coordenadas que se den correspondan al tablero. Luego que la casilla a la que se tira no esté ya ocupada por otra ficha. Por último habrá que ver si tirando en esa casilla se consigue aprisionar por lo menos una ficha del contrario, ya que si no no se puede tirar ahí. Una vez realizados todos estos controles, hay que poner la ficha en el tablero y cambiar de bando las fichas aprisionadas. Por último habrá

que actualizar los contadores de las fichas que tiene cada uno y pasarle el control a la jugada del contrario, en este caso el ordenador.

Hay que tener en cuenta el caso de que el jugador no pueda tirar porque no haya ninguna casilla en la cual pueda girar fichas del contrario. Para ello se debe prever una respuesta especial (por ejemplo «Ø») que le indique a la máquina que no puede tirar y le pase el control al siguiente proceso. Para que el jugador no haga trampas (a veces es interesante no tirar en una jugada, por ejemplo cuando sólo se puede tirar en un sitio que permitirá al contrario coger una esquina) habrá que comprobar esto antes de pasar el control. En caso de que no sea cierto se emitirá un mensaje y se volverá a pedir la jugada. Si por el contrario es cierto habrá que guardarlo en alguna variable para luego poder ver si se ha terminado la partida. Este apartado, dado que es mś complejo que el anterior, lo representaremos en un gráfico de la siguiente manera:



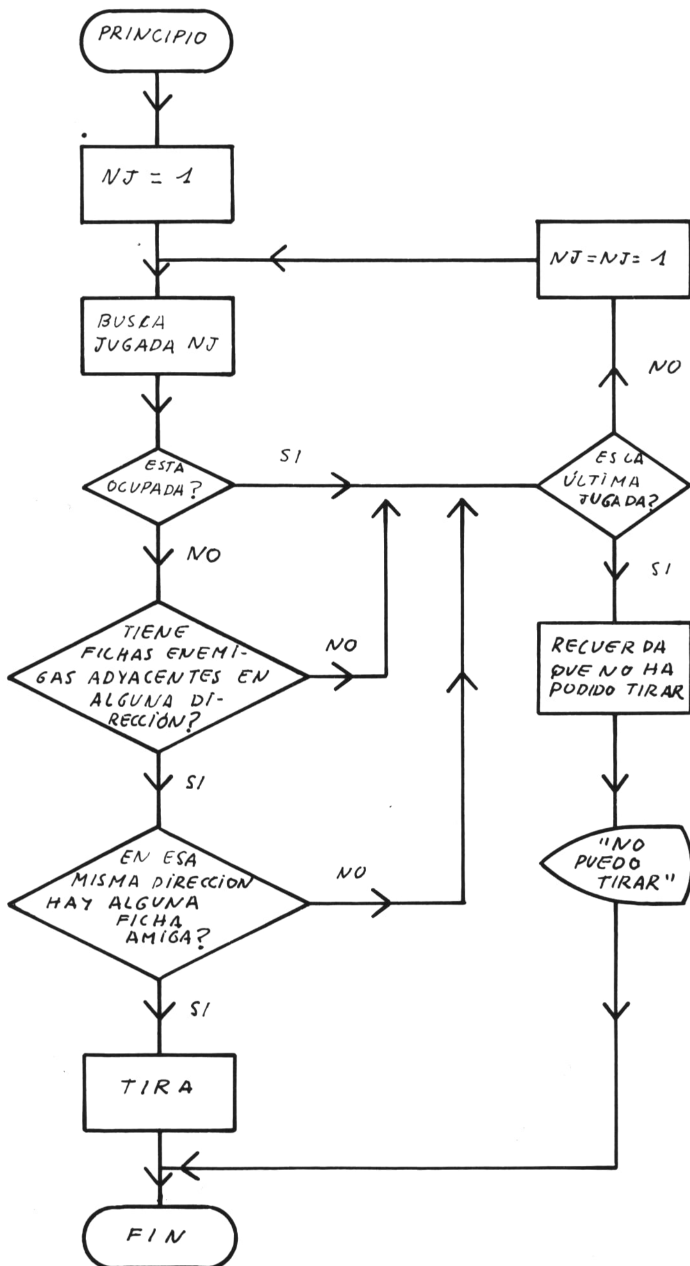
El gráfico es muy general ya que no profundiza en los procesos de hacer la tirada y de controlar si es correcta. De estos dos procesos haremos la afinación más adelante porque veremos que son comunes con la jugada del ordenador.

## ¿HA TERMINADO LA PARTIDA?

Una partida de Othello puede terminar de dos formas. La primera se da cuando el tablero está lleno y para saberlo sólo hay que mirar los contadores de fichas que se van actualizando en cada jugada comprobando que no superen el valor 64 ya que es el total de fichas que caben en el tablero. La segunda es que ninguno de los dos jugadores pueda tirar, que es muy infrecuente, para ello habrá que tener en cuenta si la jugada del tirador ha sido uno 0 (no puede tirar) y la del contrincante anterior también. Este trozo del programa es igual en las dos partes en que aparece dentro del diagrama principal.

## JUGADA DEL ORDENADOR

La jugada del ordenador es la parte central del programa. De este proceso depende que el programa juegue bien o mal. En este primer programa se hará una elección puramente posicional, de manera que el proceso quedará bastante simplificado y tardará poco tiempo. Como contrapartida tenemos que no jugará tan bien como el segundo, en el que haremos búsquedas a varios niveles. El método será el siguiente: Recordemos que en la inicialización hemos almacenado por oden (de mayor a menor) las casillas del tablero según la puntuación que les habíamos asignado previamente. Pues bien lo primero que haremos es ver si podemos tirar en la primera casilla, que es la mejor. Si no se puede probaremos con la segunda y así sucesivamente hasta que se encuentre una casilla en la que sea posible tirar. Esta casilla cumplirá la condición de que es la mejor posición (siempre según el criterio anteriormente comentado) a la que se puede tirar. Una vez se tiene la casilla sólo hay que efectuar la tirada, actualizar contadores y pasar el control al contrario. Puede darse el caso de que no se pueda tirar en ninguna de las casillas, entonces se emitirá un mensaje y se guardará un «0» como última jugada del ordenador para luego comprobar si ha terminado la partida: (NJ es la abreviación de «número de jugada»).


PRINCIPIO
NJ = 1
NJ=NJ= 1
BUSCA
JUGADA NJ
NO
SI
ESTA
OCUPADA?
ES LA
ÚLTIMA
JUGADA?
NO
SI
TIENE
FICHAS ENEMI-
GAS ADYACENTES EN
ALGUNA DI-
RECCIÓN?
NO
RECUERDA
QUE NO HA
PODIDO TIRAR
SI
EN ESA
MISMA DIRECCION
HAY ALGUNA
FICHA
AMIGA?
NO
"NO
PUEDO
TIRAR"
SI
TIRA
FIN

## FINAL DE PARTIDA

Consiste simplemente en averiguar quién ha ganado, emitir un mensaje de felicitación y terminar el programa. Este apartado tampoco es tan complicado como para hacer un gráfico.
Una decisión importante que hay que tomar antes de empezar a programar e incluso antes de escribir el pseudo-programa consiste en separar el programa principal de lo que serán las subrutinas. Para ello se debe tener en cuenta cuáles son los procesos que se repiten más o que es conveniente separar por razones de comprensión del programa. Las subrutinas se pueden estructurar en niveles de modo que una subrutina de nivel superior llama a otra de nivel más bajo. A la vista del diagrama principal no se ve ninguna subrutina, pero cuando se observan los diagramas de los procesos de las jugadas, tanto del humano como del ordenador se hace patente la necesidad de una subrutina que compruebe si una jugada es correcta. Esta subrutina se usará tanto en la jugada del ordenador, como en la del humano. A esta subrutina la llamaremos COMPRUEBA POSIBILIDAD. Como también se ha previsto la verificación de que el jugador humano no puede tirar, la subrutina COMPRUEBA POSIBILIDAD será usada por otra que la llamaremos MIRA SI ES VERDAD. Por lo tanto, COMPRUEBA POSIBILIDAD es una subrutina de segundo nviel. También se necesitará otra subrutina que ejecute la tirada ya que se usará para los dos jugadores. La llamaremos TIRA.
Ahora es el momento de pensar en cómo se solucionarán algunos problemas como por ejemplo el saber si necesitaremos una tabla interna que haga de tablero. Esto no será necesario puesto que al no hacer profundización de las jugadas el tablero lo tendremos en pantalla y para saber qué ficha está en cada casilla sólo tendremos que usar SCREEN$.
Con objeto de acortar palabras a la hora de escribir el pseudo-programa, nos será de gran utilidad dar los nombres de las variables más importantes que se vayan a utilizar:
A$: Contendrá el carácter que represente a la ficha del jugador que esté jugando en ese momento. Este carácter siempre será «0» para el humano o bien «X» para cuando juegue el ordenador.
E$: Es el carácter de la ficha del jugador enemigo en un momento determinado. Siempre lo contrario de A$.
H$(60),V$(60): Cada una de ellas contendrá sesenta números de una cifra (los ponemos en una variable de cadena porque así ahorran espacio, si no ocuparían cinco veces más). En H$ están las coordenadas horizontales y en V$ las verticales de todas las posiciones del tablero que quedan por tirar (64 menos las 4 del inicio) ordenadas de mejor a peor. Ejemplo: La quinta mejor casilla a la que se puede tirar tendrá por coordenadas X=VAL V$(5) e Y=VAL (H$(5).
X,Y: coordenadas de la casilla que se quiere jugar en un momento determinado.

X1,Y1: Coordenadas de la casilla que se está examinando en un momento determinado.
J: Variará de 1 a 8, y nos indica una dirección de tablero. Asociado a cada J tenemos 4 números: Los dos primeros nos indican qué cantidad debemos sumarle a las coordenadas de un punto para obtener la siguiente casilla que está en la dirección determinada por J. Estos dos números estarán en las variables IX e IY (incremento de X e incremento de Y). Y los dos segundos nos dan los valores límite que pueden alcanzar las coordenadas X e Y de una casilla si nos desplazamos en la dirección determinada por J. Estos dos números estarán en las variables FX y FY.
0(8,2): En esta variable, para cada dirección posible (8), tenemos almacenadas 2 coordenadas que serán las que nos marcarán hasta donde podemos girar fichas desplazándonos en esa dirección.
UJO: Valdrá 0 si en la última jugada del ordenador, éste no ha podido tirar. Servirá para averiguar si se ha dado uno de los posibles finales de partida junto con UJH.
UJH: Idéntico que la anterior pero para el jugador humano.
CONTX: Contará las fichas «X» que hay en el tablero.
CONTO: Contará las fichas «O» que hay en el tablero.
Comentarios sobre las variables: La variable 0(8,2) no tiene sentido si no están fijadas unas coordenadas a partir de las cuales empezaremos a girar fichas hasta llegar al límite que nos marca para cada dirección la variable 0(8,2).
Vamos ahora a escribir el programa en pseudolenguaje:

**PROGRAMA PRINCIPAL**

```
INICIALIZA TODO
PREGUNTA QUIEN EMPIEZA
SI EMPIEZA HUMANO VES A H:

O:--JUGADA ORDENADOR-
FICHA AMIGA="X", ENEMIGA="O"
BUSCA LA MEJOR JUGADA
SI NO HAY JUGADAS POSIBLES VES A H:
APUNTATE UNA FICHA MAS
TIRA
MIRA SI ES FINAL DE PARTIDA

H:—JUGADA DEL HUMANO—
FICHA AMIGA="O", ENEMIGA="X"
PREGUNTA COORDENADAS
SI NO PUEDE TIRAR MIRA SI ES MENTIRA Y:
SI ES MENTIRA VUELVE A H:
```

SI NO LO ES VES A O:
COMPRUEBA JUGADA
SI NO ES BUENA VES A H:
APUNTATE UNA FICHA MAS
TIRA
MIRA SI ES FINAL DE PARTIDA
—FIN—

## SUBRUTINAS

—COMPRUEBA POSIBILIDAD—
PARA CADA UNA DE LAS OCHO DIRECCIONES HAZ:
SI ESTAS AL FINAL DEL TABLERO EXAMINA OTRA DIRECCION
SI ENCUENTRAS UNA VACIA EXAMINA OTRA DIRECCION
SI ENCUENTRAS UNA ENEMIGA HAZ:
BUSCA LA PROXIMA FICHA AMIGA EN ESA DIRECCION Y
GUARDATELA PARA LUEGO.
ALMACENA QUE LA JUGADA ES BUENA.
—FIN—

—BUSCA JUGADA—
ESCOGE LA PRIMERA JUGADA DE LA LISTA
COMPRUEBA SI ES POSIBLE
SI NO ESCOGE LA SEGUNDA Y ASI SUCESIVAMENTE HASTA
QUE ENCUENTRES UNA.
—FIN—

—MIRA SI ES VERDAD—
BUSCA JUGADA
SI NO LA ENCUENTRAS ALMACENA QUE NO ES MENTIRA
SI LA ENCUENTRAS ALMACENA QUE ES MENTIRA Y DI DONDE
PUEDE TIRAR
—FIN—

—TIRA—
IMPRIME LA FICHA JUGADA EN LAS COORDENADAS
CORRESPONDIENTES
PARA CADA DIRECCION HAZ:
MIRA SI SE PUEDEN GIRAR FICHAS EN ESA DIRECCION
SI NO SE PUEDE CAMBIA DE DIRECCION
SI SE PUEDE:
GIRA LAS FICHAS HASTA ENCONTRAR UNA ENEMIGA
PARA CADA UNA QUE GIRES AUMENTA EL CONTADOR DE
LAS AMIGAS Y DISMINUYE EL DE LAS ENEMIGAS
IMPRIME LOS CONTADORES
—FIN—

—MIRA SI ES FINAL—
MIRA SI EL TABLERO ESTA LLENO Y SI LO ESTA VES A FINAL DE PARTIDA
MIRA SI NINGUNO DE LOS DOS PUEDE TIRAR Y SI ESTO OCURRE VES AL FINAL DE PARTIDA
—FIN—

—FINAL DE PARTIDA—
ENCUENTRA AL GANADOR
FELICITALO
TERMINA EL PROGRAMA
—FIN—

—INICIALIZACION—
DAR LOS VALORES A LAS CASILLAS
DIBUJAR EL TABLERO CON SU POSICION INICIAL
INICIALIZAR EL CONTADOR DE JUGADAS
INICIALIZAR LOS CONTADORES DE PUNTOS
—FIN—

A la vista del programa en pseudo-lenguaje, ya se puede pasar a escribir el programa en el BASIC del Spectrum. El listado del programa contine muchas líneas de comentario (setencias REM) para facilitar su comprensión y su relación con el pseudo-programa:

```
  10 GO SUB 9000: REM INICIALIZA
CION
  20 INPUT "QUIERE EMPEZAR? (S/N
)   "; LINE r$
  30 IF r$="S" OR r$="s" THEN GO
 TO 1000
  35 REM
  36 REM ************************
  37 REM  TIRADA DEL ORDENADOR
  38 REM ************************
  39 REM
  40 LET A$="O": LET E$="X"
  50 LET PR=1
  60 LET X=VAL M$(1,PR): LET Y=V
AL M$(2,PR)
  70 GO SUB 8000
  80 IF OK=0 AND PR<60 THEN LET
PR=PR+1: GO TO 60
  82 IF OK=0 AND PR=60 THEN PRIN
T AT 20,0;"NO PUEDO TIRAR, HAZLO
 TU": BEEP 3,-5: PRINT AT 20,0;"
                               ":
GO TO 1000
  85 LET contb=contb+1
  90 GO SUB 7000
 100 IF contn+contb=64 THEN GO S
UB 9500
 990 REM
 991 REM ************************
 992 REM   TIRADA DEL JUGADOR
 993 REM ************************
```

```
1000 LET A$="X": LET E$="O"
1010 INPUT "VERTICAL:";X
1015 IF X=0 THEN GO TO 40
1020 INPUT "HORIZONTAL:";Y
1030 PRINT AT 20,0;"
"
1040 GO SUB 8000
1050 IF OK=0 THEN PRINT AT 20,0;
"  JUGADA ERRONEA": BEEP 2,-10:
GO TO 1010
1055 LET contn=contn+1
1060 GO SUB 7000
1065 IF contn+contb=64 THEN GO S
UB 9500
1070 GO TO 40
6990 REM
6991 REM ************************
6992 REM  EJECUCION DE LA TIRADA
6993 REM ************************
6994 REM
7000 RESTORE 8010
7010 PRINT AT x,y;a$
7020 FOR j=1 TO 8
7030 READ ix,iy,fx,fy
7040 IF o(j,1)=0 THEN GO TO 7200
7045 LET X1=X: LET Y1=Y
7050 LET x1=x1+ix: LET y1=y1+iy
7060 IF x1=o(j,1) AND y1=o(j,2)
THEN GO TO 7200
7070 BEEP .1,25: PRINT AT x1,y1;
a$
7074 IF A$="O" THEN LET contb=co
ntb+1: LET contn=contn-1
7076 IF A$="X" THEN LET contn=co
ntn+1: LET contb=contb-1
7100 PRINT AT 9,22;"  ";AT 9,22;
contb
7110 PRINT AT 11,22;"  ";AT 11,2
2;contn
7120 GO TO 7050
7200 NEXT j
7300 RETURN
7990 REM
7991 REM ************************
7992 REM COMPROBACION POSIBLIDAD
7993 REM ************************
7994 REM
8000 RESTORE 8010
8010 DATA 1,0,8,0
8011 DATA 0,1,0,8
8012 DATA 0,-1,0,1
8013 DATA -1,0,1,0
8014 DATA 1,1,8,8
8015 DATA -1,-1,1,1
8016 DATA 1,-1,8,1
8017 DATA -1,1,1,8
8020 DIM o(8,2): LET ok=0
8030 FOR j=1 TO 8
8040 READ ix,iy,fx,fy
8050 IF x=fx OR y=fy THEN GO TO
8200
8055 IF SCREEN$ (X,Y)<>" " THEN
GO TO 8200
8060 LET x1=x+ix: LET y1=y+iy
```

```
8070 IF SCREEN$ (x1,y1)=e$ THEN
GO SUB 8300
8200 NEXT j
8250 RETURN
8300 LET x1=x1+ix: LET y1=y1+iy
8310 IF SCREEN$ (x1,y1)="█" THEN
 RETURN
8315 IF SCREEN$ (x1,y1)=" " THEN
 RETURN
8320 IF SCREEN$ (x1,y1)=a$ THEN
LET ok=1: LET o(j,1)=x1: LET o(j
,2)=Y1: RETURN
8330 GO TO 8300
8990 REM
8991 REM ************************
8992 REM      INICIALIZACION
8993 REM ************************
8994 REM
9000 PRINT AT 0,0;"██████████"
9010 FOR i=1 TO 8: PRINT AT i,0;
"█        █";i: NEXT i
9020 PRINT AT 9,0;"██████████"
9030 PRINT AT 10,1;"12345678"
9040 PRINT AT 4,4;"OX";AT 5,4;"X
O"
9045 DIM m$(2,60)
9050 LET m$(1)="188161386138633б
4158154365643534645722327742б357
712182877272"
9060 LET m$(2)="881116168383363б
84845115435346562775367436257224
128771282772"
9080 LET contb=2: LET contn=2
9090 PRINT AT 9,19;"YO:";contb
9100 PRINT AT 11,19;"TU:";contn
9110 PRINT AT 4,4;"OX"
9120 PRINT AT 5,4;"XO"
9130 RETURN
9490 REM
9500 REM ************************
9510 REM    FINAL DE PARTIDA ?
9520 REM ************************
9525 REM
9530 IF contn+contb<>64 THEN RET
URN
9540 IF contn>contb THEN PRINT A
T 17,0;"MUY BIEN, HAS GANADO": B
EEP 10,0: STOP
9550 IF contn<contb THEN PRINT A
T 17,0;"LASTIMA, HAS PERDIDO": B
EEP 5,-30: STOP
9990 FOR l=1 TO 8: PRINT AT l,20
;o(l,1);o(l,2): NEXT l
```

# COMENTARIOS SOBRE EL PROGRAMA DEL OTHELLO

La variable de cadena m$ tiene dimensiones 2×60 y en ella están contenidas las casillas del tablero. En m$(1) están las primeras coordenadas, y en m$(2) las segundas coordenadas. El orden en el que están corresponde al orden de su puntuación, de mejor a peor. Las dos subrutinas más importantes son las llamadas «COMPROBACION POSIBILIDAD» y «EJECUCION DE LA TIRADA». La subrutina «COMPROBACION POSIBILIDAD» empieza en la línea 8000 y realiza las siguientes funciones:

— Comprueba que se pueda tirar en la posición que viene determinada por las coordenadas que están en ese momento en las variables X e Y.
— Actualiza la variable O(8,2) con unos valores que le indicarán a la subrutina «EJECUCION DE LA TIRADA» hasta qué ficha tiene que girar para una dirección determinada.
— En caso de que se pueda tirar coloca el valor 1 en la variable OK.

Esta subrutina es muy independiente y para realizar estas funciones hace los siguientes pasos: En primer lugar (línea 8020) coloca la variable OK a 0 así como la variable 0(8,2) ya que al ser dimensionada en este punto del programa su valor es todo ceros. Para comprobar que se puede tirar en una casilla determinada, tiene que ver que al menos para una de las direcciones puede girar fichas enemigas. Para ello se hace un examen para cada dirección, esto es, un bucle cuyo principio está en la línea 8030 y su final en la línea 8200. En este bucle, la variable J nos indica la dirección en la que que estamos mirando, que viene determinada por los valores que se han leído en la línea 8040. Estos valores, que están en las sentencias DATA de las líneas 8010 a 8017 son 4 para cada dirección y son: ix, que nos indica en cuánto debemos incrementar la primera coordenada para movernos una casilla en esa dirección, iy que es el incremento de la segunda coordenada; fx, que nos indica qué valor toma la primera coordenada cuando hemos llegado al final del tablero en esa dirección y por último fy, que es lo mismo para la segunda coordenada. Una vez leídos los valores, se mira si estamos en un extremo del tablero para esa direción (línea 8050), y si esto ocurre, ya podemos pasar a examinar otra dirección. Luego se mira si la casilla ya está ocupada (línea 8055). Seguidamente se avanza

una casilla en esa dirección (línea 8060), y se comprueba que haya una ficha contraria (línea 8070). Si no la hay podemos pasar a la siguiente dirección (línea 8200), pero si la hay, debemos examinar si en la misma dirección, pero más adelante, encontramos otra ficha de nuestro bando pero sin ningún cuadrado vacío entremedio. Esto se hace en la subrutina que empieza en la línea 8300. En ella se avanza otra casilla (línea 8300) y se comprueba que no estemos al final del tablero (línea 8310), que no esté vacía (línea 8315) y por último, si encontramos una ficha de nuestro bando, colocamos un 1 en la variable OK conforme la casilla es correcta y en la variable O las coordenadas de la casilla en que se encuentra esa ficha de nuestro bando.
Esta rutina es quizás la más complicada del programa e incluso se puede mejorar para ganar tiempo. Por ejemplo, se puede colocar la línea 8055 fuera del bucle, ya que si la casilla está ocupada no hace falta mirarlo 8 veces.
Para comprender las líneas DATA más claramente obsérvese el gráfico que viene a continuación en el que aparecen las casillas numeradas según la posición PRINT que ocupan en la pantalla:

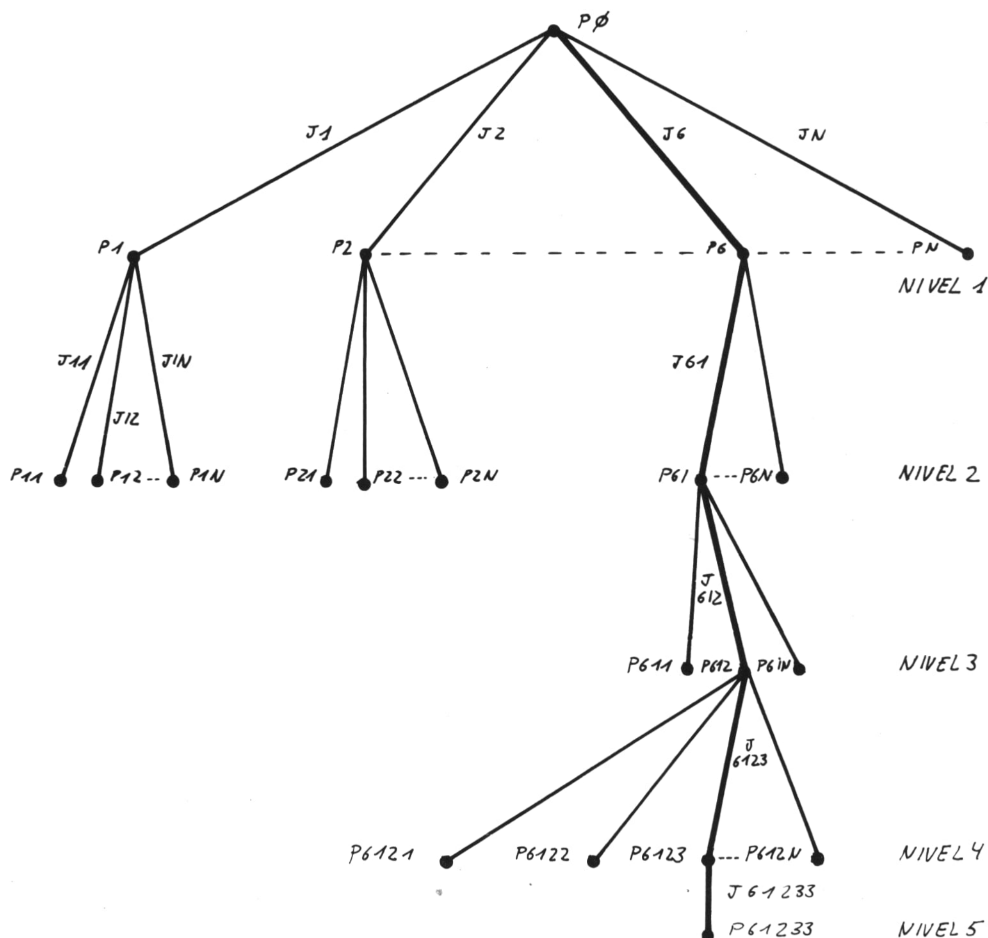
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 |
| 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 |

Observemos las flechas en diagonal que nos indican una dirección a examinar. La línea DATA que corresponde a esa dirección es la 8016 que contiene los número 1,−1,8,1 ya que para avanzar en esa dirección tenemos que sumarle 1 a la primera coordenada y restarle 1 a la segunda. Y además siempre que se llega al final del tablero en esa dirección encontramos un 1 en la primera coordenada O un 8 en la segunda. Quiero remarcar que se llega al final tanto si se encuentra un 1 en la primera coordenada como si se encuentra un 8 en la segunda, por eso en la línea 8050 tenemos un OR y no un AND y las dos condiciones a la vez sólo se cumplirían en el caso de llegar a la esquina. También por eso cuando las direcciones no son

diagonales alguno de los dos últimos datos contiene un 0 ya que no importa para nada saber si hemos llegado al final.
La subrutina «EJECUCION DE LA TIRADA» recoge los datos de la variable O para saber qué fichas tienen que girar. Actúa del siguiente modo: Primero imprime en la casilla correspondiente una ficha del bando que juega (a$), y luego realiza, al igual que la otra, un examen para cada dirección utilizando las mismas líneas DATA, aunque no usa las dos últimas variables y mira en la matriz o en las variables correspondientes a esa dirección, para ver si hay que girar alguna ficha (línea 7040) si no hay que hacerlo, pasa a examinar otra dirección. En caso contrario va avanzando casillas en esa dirección y girándolas (línea 7070) hasta que llega a la última que hay que girar, que es la que está en la matriz o, en la fila correspondiente a la dirección (línea 7060). Por cada ficha girada incrementa o decrementa los contadores correspondientes y luego los imprime en pantalla (línea 7074 a 7110).
En este programa faltan algunos filtros a las respuestas e incluso a las jugadas, pues habría que comprobar que cuando el oponente al ordenador introduce un 0 como respuesta está diciendo la verdad, es decir, que realmente no puede tirar. El siguiente programa contiene estos filtros y supongo que no le costará mucho al lector añadirlos a éste.

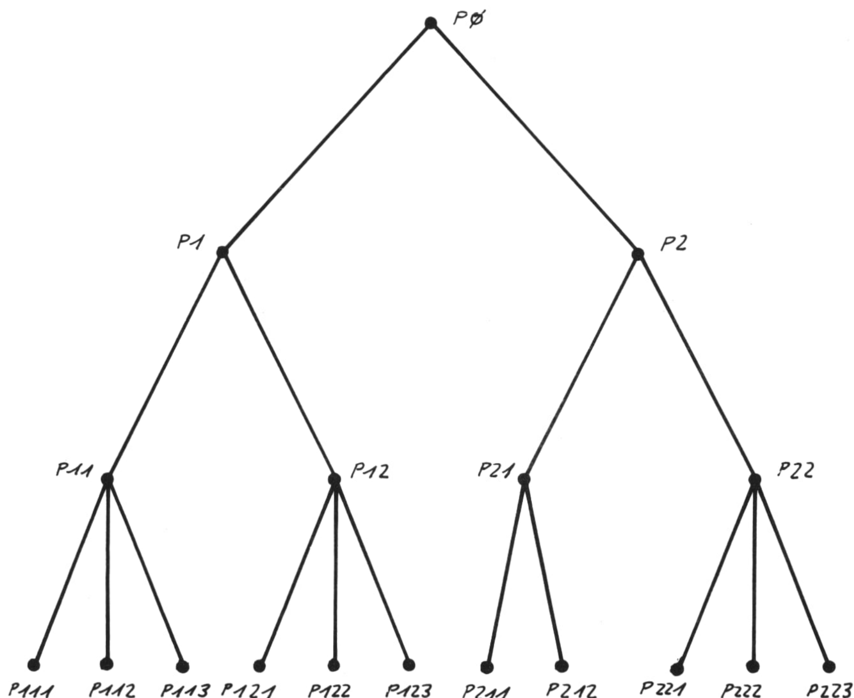
# EXAMINANDO EN PROFUNDIDAD: BUSQUEDA POR NIVELES

Si usted no había jugado antes al Othello, quizás le habrá costado un poco ganar al programa anterior, pero estoy seguro de que tarde o temprano lo habrá conseguido, y más, conociendo el método que se sigue para la elección de las jugadas. El principal defecto del programa es que no tiene en cuenta sus posibles respuestas y aunque él escoja una buena posición, quizás con otra jugada, habría evitado una buena posición del adversario. Otro aspecto que no se tiene en cuenta es el del número de fichas que se ganan en una jugada, pero está claro, que con esta estructura no se puede tener en cuenta pues aunque en una jugada consiga girar muchas fichas, el adversario las puede volver de su color en la jugada siguiente. Con esto llegamos a un punto en el que se tiene la imperiosa necesidad de examinar las posibles respuestas del contrario a una jugada del ordenador. Este libro no sería suficiente ni siquiera para dar una visión general de lo mucho que se ha profundizado en el tema pero se intentará explicar un método que nos ayude a ello. Lo primero que vemos es que a una jugada del ordenador le pueden seguir muchas del adversario y cada una de ellas puede tener a su vez muchas respuestas del ordenador. Esto se da en todos los juegos de dos personas. La profundidad con la que se investiga una jugada se llama nivel. Es decir, si nosotros analizamos las respuestas posibles a cada una de las jugadas que podemos hacer en un momento determinado, estaremos examinando a nivel 2. Esto se ve mejor si representamos las jugadas en un árbol en el que las ramas son las jugadas, y las posiciones obtenidas son los nudos:

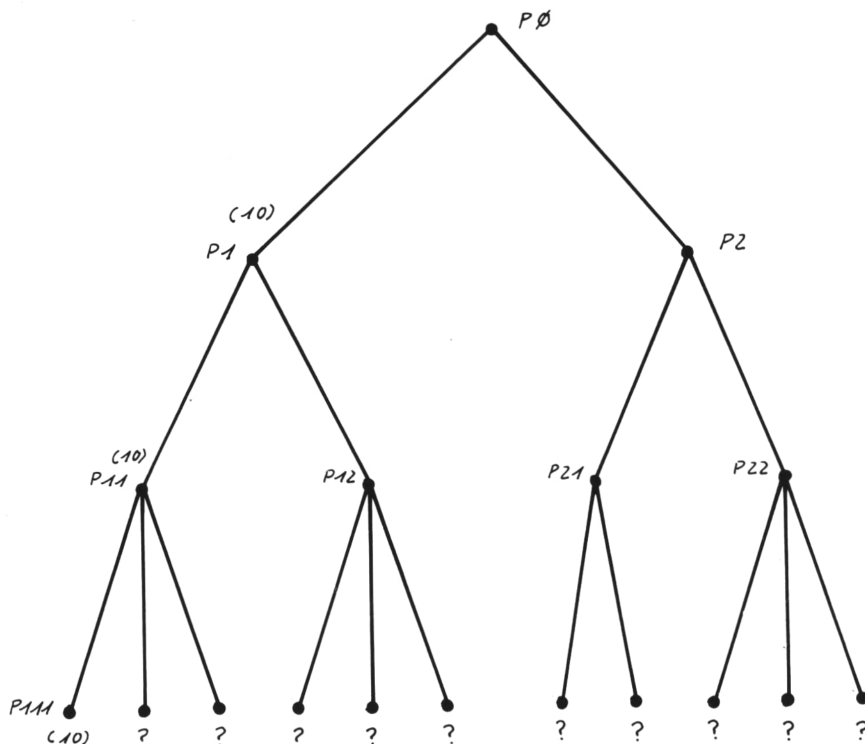



Como se ve en el dibujo, partiendo de una posición inicial P0 podemos realizar varias jugadas J1, J2,........JN que se corresponden con las ramas que parte de P0. Cada una de estas jugadas nos hace llegar a una posición. Así la jugada J1 nos lleva a la posición P1, la jugada J2 a la posición P2, etc... A su vez, desde la posición P1 el contrario puede responder con varias jugadas, la J11, la J12..., que nos llevan a las posiciones P11, P12, etc., y desde la posición P2 puede realizar otras jugadas, cuyo número puede ser distinto, J21, J22, etc., que nos llevan a otras tantas posiciones P21, P22, P23... Esta notación simplifica mucho el proceso ya que si en un momento determinado, estamos refiriéndonos por ejemplo a la posición P61233 ya sabemos que es una posición de quinto nivel, ya que a la P le siguen 5 números y que se llega a ella después de haber hecho nuestra jugada J61233, en respuesta a su jugada J6123, que a su vez respondía a nuestra jugada J612 que se efectuó contestan-

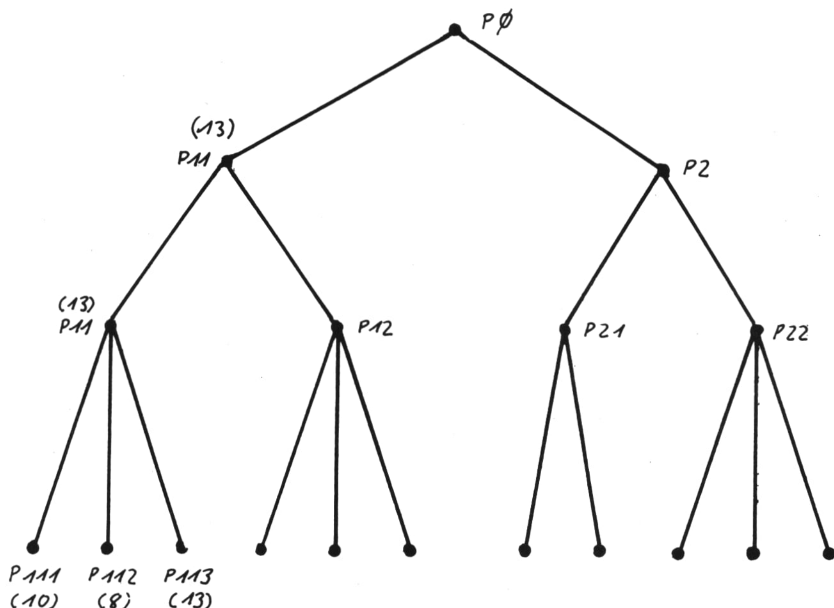
do a su J61, que respondía a nuestra J6 y que ésta partía de la posición P0.
Lo que nosotros buscamos es cuál de las posibles jugadas de nivel uno es mejor para responder en un momento determinado del juego. Para ello se puede asignar una puntuación a cada posición. Esta puntuación se asignará siguiendo a su vez un método muy estricto. La forma de evaluar una posición depende de cada juego en particular, por ejemplo en el caso del Othello, se tendrá en cuenta el lugar que ocupa la ficha tirada (si es bueno o no lo es según el criterio ya explicado) y también el número de fichas que se giran. Esto no es exactamente dar un valor a la posición, sino dar un valor a la jugada, aunque ambas cosas estén íntimamente relacionadas.
En el programa de Othello que se explica más adelante, se puntúan las jugadas, y no las posiciones. Para puntuar las posiciones habría que contar para cada una las fichas que hay de cada color en un momento determinado y no las que se han girado, también habría que puntuar las posiciones de todas las fichas del tablero y no sólo la de la ficha que se tira. Pero esto lo haría aún más lento. Aclarado esto seguiremos explicándolo como si se puntuara a la posición y no a la jugada. Para comprender el método lo haremos basándonos en un ejemplo abstracto de tres niveles de juego. Observe para ello el siguiente árbol:



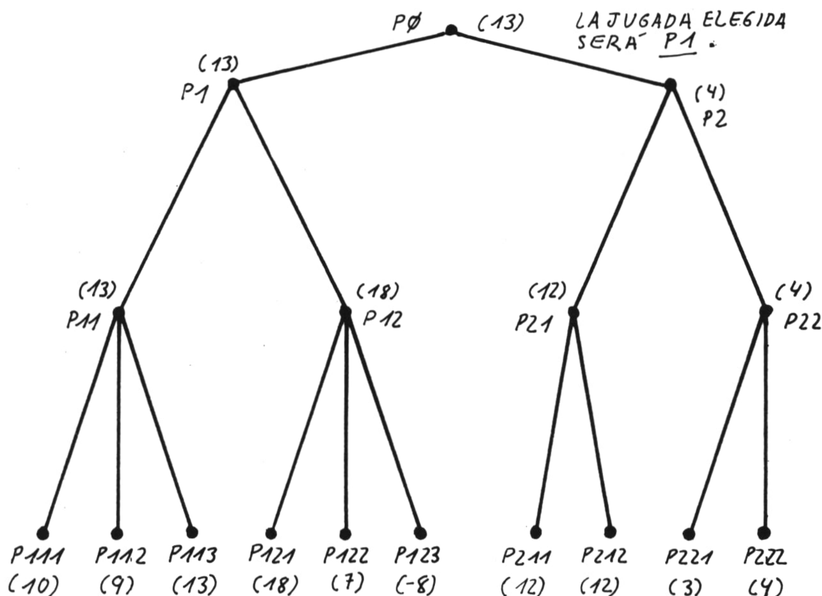
En este momento estamos en la posición P0. Lo primero que haremos será copiar el tablero real en otro de pruebas para poder realizar las valoraciones. Una vez hecho esto, se juega ya en el tablero de pruebas la primera jugada J1, su primera respuesta J11 y la primera respuesta a ella: la J111 una vez en este punto se evalúa la posición P111 a la que se ha llegado después de realizadas las jugadas y se obtienen por ejemplo 10 puntos. Con ello tenemos el árbol de la siguiente manera (al lado de la posición se ponen los puntos obtenidos).



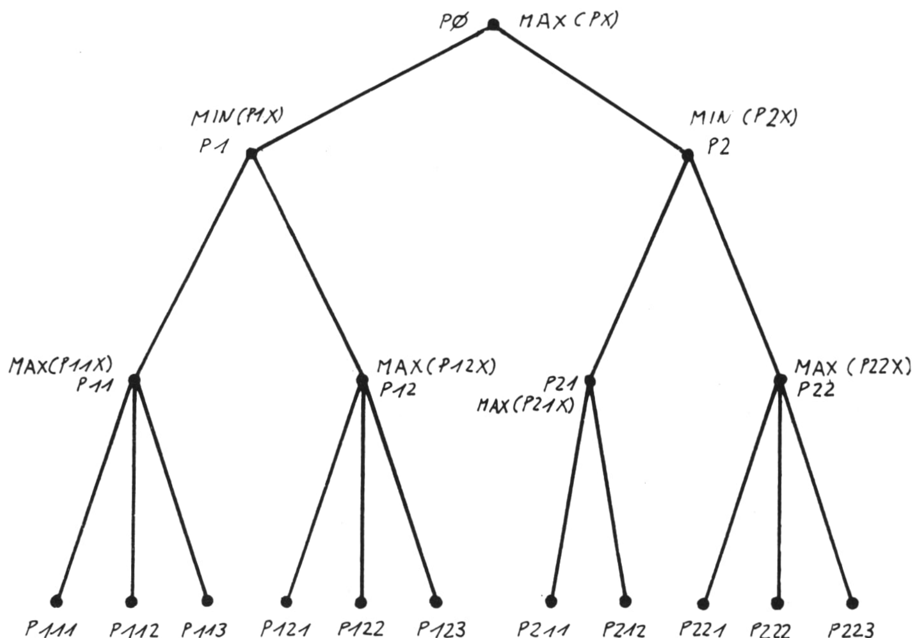

Como no se han examinado más jugadas, de momento ésta es la puntuación que obtendremos tanto para P11 como para P1. Pero luego examinamos lo que pasaría si a partir de la posición P11 se efectuara la jugada J112 evaluando la posición P112 y suponemos que nos da 8 puntos. Dado que las jugadas del tercer nivel las realiza la máquina y no el jugador humano nos quedaremos con la P111 y por tanto no variarán las puntuaciones de P11 y de P1 en el árbol, ya que siempre realizaremos la mejor jugada:




Si luego al examinar J113 nos da por ejemplo 13 puntos en P11:
entonces J113 será la jugada escogida hasta el momento, y coloca
remos la puntuación de P113 en P11 y en P1. En definitiva, en P11
se coloca el máximo obtenido de todas las posiciones del tipo P11X
cualquiera que sea X. Así después de haber sido examinadas todas
las jugadas del tipo J11X el árbol tendría una forma así:



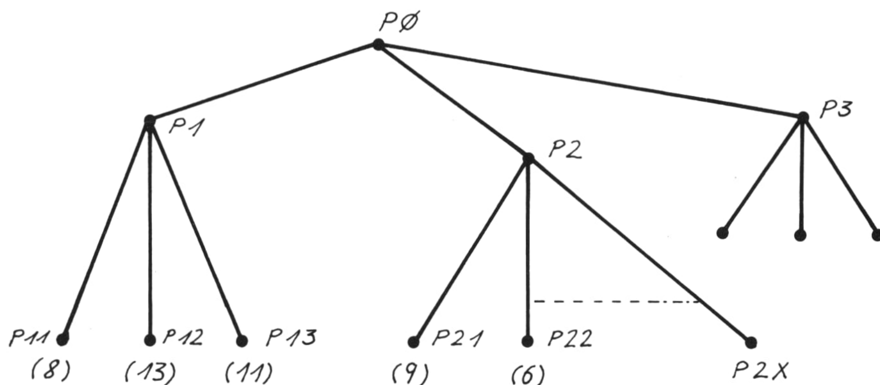
Después recuperamos el tablero en la posición P1 y realizamos la jugada J12, examinando todas las respuestas posibles y puntuando las posiciones P121, P122, etc. y en P12 colocaremos el máximo de los P12X. La diferencia está en que ahora, en P1 no pondremos el máximo de P11 y P12 sino el mínimo, ya que las jugadas J11 y J12 las realiza el contrario y por tanto escogerá la que dé una puntuación más baja, suponiendo que sea un buen jugador. En general, el árbol quedaría estructurado de la siguiente manera:




La notación MAX(PX) significa el máximo de P11, P12, P13, etc., y MIN(P2K) significa el mínimo de P21, P22, P23, etc.

El método tiene un problema, y es la gran cantidad de valoraciones que hay que hacer. Para tener una idea hagamos unos cálculos: el 99 % de las veces, el Othello termina habiéndose jugado las 60 fichas. Vamos a suponer que para cada jugada sólo hubiera 3 posibles respuestas, aunque son más. Si quisiéramos examinar a nivel 60 tendríamos que hacer al principio de la partida 4 por 10 elevado a 28 valoraciones aproximadamente (es decir, un 4 seguido de 28 ceros aunque en realidad serían muchas más). Suponiendo que cada una tardara 1 microsegundo (que es muy poco, ya que la mayoría de las instrucciones del Z80 ya tardán más) tendríamos que esperar 4E22 segundos, más de 40 billones de años. En fin, que dudo que alguien se espere a que conteste el ordenador. Debido a esto, en los juegos con gran crecimiento como el Othello, las damas, el ajedrez, etc., no se puede investigar hasta el final, y

se escoge un nivel que dé un tiempo de respuesta aceptable. Aparte de esto, el método se puede refinar para que no se tengan que examinar todas las jugadas. Veamos un ejemplo: Supongamos que estamos haciendo una profundización a 2 niveles y que una vez examinadas unas cuantas jugadas, el árbol presenta el siguiente estado:




Observamos que el mínimo de los P1X es 8 y que en la jugada J22 se ha obtenido una puntuación de 6 por lo tanto el mínimo de los P2X siempre será menor que 6, que a su vez es menor que 8, por lo tanto nunca será escogida la jugada J2 por lo que ya no nos hace falta para nada seguir examinando las jugadas J23, J24, etc. que pueden ser muchas. Con esto se ve, que el resultado que se obtenga en P0 1 final será el mismo que si se hubieran examinado todas, pero con un ahorro considerable de tiempo dado que podemos desechar muchas valoraciones.

Si además, el orden en que se eligen las jugadas a examinar de cada nivel, es de mejor a peor, entonces es más posible que nos aparezcan primero los valores más adecuados en cada nivel para desechar los siguientes exámenes de las demás ramas que parten de ese nudo. Es evidente que no sabemos cuál es la mejor jugada, pero lo que sí tenemos es una idea. Por ejemplo, está claro que el orden que se seguirá en el Othello es el que se ha dado a las jugadas en el primer programa.

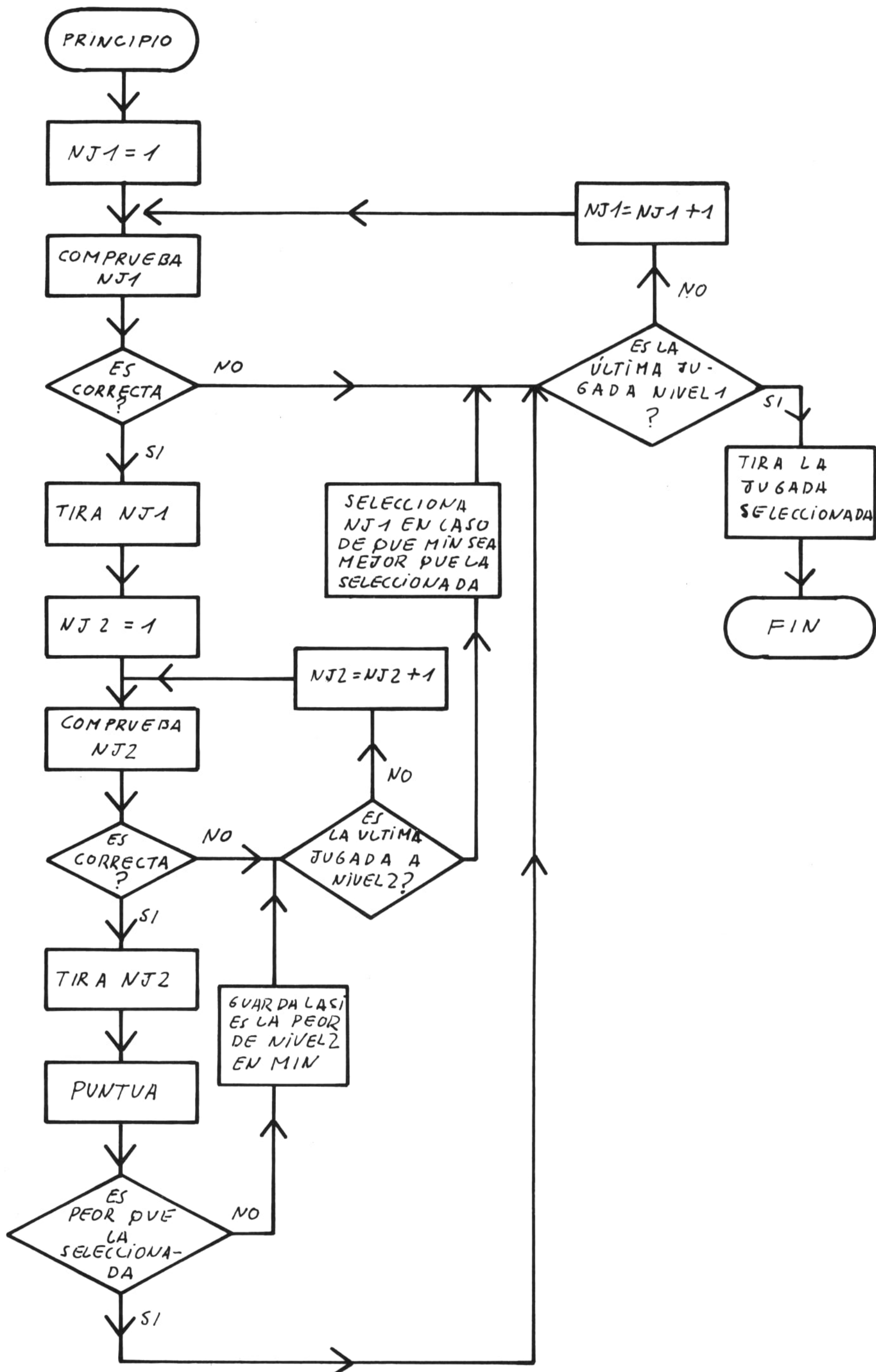
El siguiente programa es de nuevo nuestro amigo el Othello, esta vez examinado a 2 niveles de profundidad, en él se aplica todo lo que se acaba de explicar, aun así tarda una media de 4 minutos en responder a una jugada ¡y eso que sólo está a nivel 2! Pero esto es debido a la lentitud del BASIC interpretado ya que en código máquina no tardaría más de 2 o 3 segundos. Pero lo que interesa es que el programa sea comprensible.

# EL OTHELLO A NIVEL 2

Este programa es igual que el otro excepto en un punto del algoritmo: Se trata del proceso que llamábamos TIRADA DEL ORDENADOR. Por eso supondremos que el diagrama es igual que en el otro y sólo haremos el refinamiento y el pseudoprograma de esta parte. Para ello necesitamos introducir una nueva variable que nos diga los puntos que tiene cada casilla, y definir exactamente la función que nos evaluará los puntos de la jugada. También pondremos un filtro a la respuesta del jugador por si es mentira que no puede tirar, diciéndole además, una de las casillas en que puede hacerlo. También se prevé un caso en el que ninguno de los dos jugadores pueda tirar, que no estaba previsto en el otro programa.
En el gráfico ya refinado del proceso «TIRADA DEL ORDENADOR» (página siguiente) representaremos por NJ1 el número de jugada de primer nivel, y por NJ2 el número de jugada de segundo nivel.

Las subrutinas principales del programa anterior sirven para éste pero hay que duplicarlas para que examinen y tiren en un tablero interno del programa (que estará en la variable T$) ya que si lo hicieran en la pantalla, destrozarían la posición actual.
Un truco para programar este método consiste en darle a los niveles impares (que son los que su puntuación se obtiene calculando el máximo del nivel inferior) el valor más pequeño posible, mientras que a las posiciones de nivel par se les asigna el valor más grande posible, ya que se obtienen calculando el mínimo. Para que el programa en pseudolenguaje nos ocupe menos espacio vamos a relacionar algunas variables:

— SW. Al igual que OK es un «switch» o interruptor, que nos indica si se ha validado alguna de las jugadas de primer nivel.
— MAX. Contiene la puntuación de la posición P0 en un momento determinado, se llama MAX ya que es el máximo de los MIN.
— MIN. Contiene el mínimo de las puntuaciones de las jugadas de segundo nviel que se han examinado hasta el momento.
— JU. Contiene el número de jugada cuya puntuación está en MAX, es decir, la mejor jugada hasta el momento.
— A. Contiene la jugada de primer nivel que corresponde a las jugadas de segundo nivel que se están examinando en un momento determinado. Su valor pasa a JU cuando el mínimo de las puntua-


PRINCIPIO
NJ1 = 1
COMPRUEBA NJ1
ES CORRECTA ?
NO
SI
TIRA NJ1
NJ 2 = 1
COMPRUEBA NJ2
ES CORRECTA ?
NO
SI
TIRA NJ2
PUNTUA
ES PEOR QUE LA SELECCIONA-DA
NO
SI
GUARDA LA SI ES LA PEOR DE NIVEL2 EN MIN
ES LA ULTIMA JUGADA A NIVEL2?
NO
NJ2 = NJ2 + 1
SELECCIONA NJ1 EN CASO DE QUE MIN SEA MEJOR QUE LA SELECCIONADA
ES LA ÚLTIMA JUGADA NIVEL1 ?
NO
SI
NJ1 = NJ1 + 1
TIRA LA JUGADA SELECCIONADA
FIN

ciones de estas jugadas es mayor que MAX, por lo tanto es una jugada mejor.

— PH. Contiene la puntuación de la jugada de segundo nivel que se está examinando. Su valor se asigna mediante una fórmula que tiene en cuenta la posición de la ficha tirada, así como el número de fichas girados.

— PO. Contiene la puntuación de la jugada de primer nivel que se está examinando. Se asigna igual que PH.

— FG. Contiene el número de las fichas giradas en una jugada determinada. Sirve para calcular PO y PH.

— P$. Contiene los puntos de las 60 posiciones del tablero.

— T$ y B$. Tienen dimensiones (8,8) y son tableros provisionales para hacer las jugadas internas y valorarlas.

— MENTIRA. Contiene un 1 si el jugador puede tirar.

Ahora procederemos a escribir el pseudoprograma del proceso «TIRADA DEL ORDENADOR».

```
—TIRADA DEL ORDENADOR—
MAX=−10000 SW=0
COPIA DEL TABLERO ORIGINAL
BJ: BUSCA SIGUIENTE JUGADA (PR)
SI LAS HAS MIRADO TODAS Y NO PUEDES TIRAR, QUE TIRE EL
JUGADOR
SI LAS HAS MIRADO TODAS EJECUTA LA MEJOR DE ELLAS
SW=1 (PUEDES TIRAR EN ALGUNA CASILLA)
GUARDATE LA JUGADA (A=PR)
TIRALA EN EL TABLERO DE PRUEBAS
COPIA ESTE TABLERO EN OTRO
PUNTUA LA JUGADA
MIN=10000
BH: BUSCA LA SIGUIENTE RESPUESTA A LA JUGADA A
SI LAS HAS MIRADO TODAS Y NO PUEDES TIRAR MIN=0 VES A
P:
TIRA LA JUGADA EN EL TABLERO DE PRUEBAS
PUNTUALA
P: SI LA PUNTUACION ES PEOR QUE MAX VES A BJ:
(ABANDONA ESTA RAMA)
MIRA SI ES MINIMA Y GUARDALA EN MIN
SI NO HAY MAS JUGADAS (DE SEGUNDO NIVEL) VES A F:
VES A BH:
F: SI EL MINIMO ES MEJOR QUE MAX, GUARDATE ESTE MINIMO
EN MAX Y GUARDATE LA JUGADA EN JU
RECUPERA EL PUNTERO DE LA JUGADA DE PRIMER NIVEL
ANTERIOR (PR=A)
SI NO ES LA ULTIMA VES A BJ:
TIRA LA JUGADA JU
—FIN—
```

Como se puede ver, la situación es un poco más complicada que en la rutina del programa anterior. También se puede observar que tardará bastante más tiempo. En cualquier caso este programa juega mejor que el otro, si no pruebe a enfrentarlos. Veamos como queda el listado del programa:

```
   1 REM ----------------------
   2 REM **********************
   3 REM    PROGRAMA PRINCIPAL
   5 REM **********************
   6 REM ----------------------
  10 GO SUB 9000: REM INICIALIZA
CION
  20 INPUT "QUIERE EMPEZAR? (S/N
)  "; LINE r$
  30 IF r$="S" OR r$="s" THEN GO
 TO 1000
  35 REM
  36 REM **********************
  37 REM  TIRADA DEL ORDENADOR
  39 REM **********************
  40 REM
  41 LET SW=0: LET MAX=-10000: L
ET JU=0: LET PR=1
  43 LET A$="X": LET E$="O"
  45 GO SUB 5000
  50 GO SUB 4500: REM  BUSCA JUG
ADA
  60 IF OK=0 AND PR=60 AND SW=0
THEN PRINT AT 20,0;"NO PUEDO TIR
AR, HAZLO TU": LET UJO=0: BEEP 3
,-5: PRINT AT 20,0;"
                    ": GO TO 1000
  65 IF OK=0 AND PR=60 THEN GO T
O 500
  70 LET SW=1: LET A=PR
  90 GO SUB 7500
  92 GO SUB 5200
 100 LET PO=VAL P$(PR)+FG*0.028*
(CONTO+CONTX)
 105 LET MIN=10000
 110 LET A$="O": LET E$="X"
 120 LET PR=1
 130 GO SUB 5100
 132 GO SUB 4500
 140 IF OK=0 AND PR=60 THEN LET
PH=0: GO TO 200
 150 GO SUB 7500
 160 LET PH=VAL P$(PR)+FG*0.028*
(CONTO+CONTX)
 200 IF PO-PH<=MAX THEN GO TO 40
0
 210 IF PO-PH<MIN THEN LET MIN=P
O-PH
 220 IF PR=60 THEN GO TO 300
 230 LET PR=PR+1: GO TO 130
 300 IF MIN>MAX THEN LET MAX=MIN
: LET JU=A
 400 LET PR=A
 410 IF PR=60 THEN GO TO 500
```

```
 420 LET PR=PR+1: GO TO 43
 500 LET A$="X": LET E$="O"
 510 LET X=VAL V$(JU): LET Y=VAL
 H$(JU)
 520 GO SUB 8000
 530 LET UJO=1: LET CONTX=CONTX+
1
 540 GO SUB 7000
 550 GO SUB 3000
 990 REM
 991 REM ***********************
 992 REM   TIRADA DEL JUGADOR
 993 REM ***********************
1000 LET A$="O": LET E$="X"
1010 INPUT "VERTICAL:";X
1013 LET MENTIRA=2
1015 IF X=0 THEN GO SUB 2000: RE
M COMPRUEBA IMPOSIBILIDAD
1020 IF MENTIRA=1 THEN GO TO 100
0
1023 IF MENTIRA=0 THEN GO TO 106
5
1025 INPUT "HORIZONTAL:";Y
1030 PRINT AT 20,0;"
                       "
1040 GO SUB 8000: REM COMPRUEBA
1050 IF OK=0 THEN PRINT AT 20,0;
"   JUGADA ERRONEA": BEEP 2,-10:
GO TO 1010
1055 LET CONTO=CONTO+1
1060 GO SUB 7000: REM TIRA
1065 GO SUB 3000: REM MIRA SI ES
 FINAL
1067 LET UJH=1
1070 GO TO 38
1900 REM
1920 REM
1930 REM       SUBRUTINAS
2000 REM
2010 REM ***********************
2020 REM     MIRA SI ES VERDAD
2030 REM ***********************
2040 REM
2050 GO SUB 4000: REM BUSCA JUGA
DA
2060 IF OK=0 THEN LET MENTIRA=0:
 RETURN
2065 LET UJH=0
2070 LET MENTIRA=1
2080 PRINT AT 20,0; FLASH 1;"MEN
TIRA, PUEDES TIRAR EN ";X;",";Y
2090 BEEP 4,-20: PRINT AT 20,0;"
                       "
2100 RETURN
2500 LET MAX=-10000: LET JU=0
2510 LET A$="X": LET E$="O"
2520 LET PR=1
2530 GO SUB 4010
2900 REM
2910 REM ***********************
2920 REM     MIRA SI ES FINAL
2930 REM ***********************
2940 REM
3000 IF CONTO+CONTX=64 THEN GO T
```

```
0 9500
3010 IF UJO=0 AND UJH=0 THEN GO
TO 9500
3020 RETURN
3900 REM
3910 REM ***********************
3920 REM      BUSCA JUGADA
3930 REM ***********************
3940 REM
4000 LET PR=1
4010 LET X=VAL H$(PR): LET Y=VAL
 V$(PR)
4015 PRINT AT 0,31; OVER 1;"*"
4020 GO SUB 8000: REM COMPRUEBA
4030 IF OK=0 AND PR<60 THEN LET
PR=PR+1: GO TO 4010
4040 RETURN
4400 REM
4410 REM ************************
4420 REM     BUSCA JUGADA 2
4430 REM ************************
4440 REM
4500 LET X=VAL V$(PR): LET Y=VAL
 H$(PR)
4505 PRINT AT 13,10;"PR:";PR
4510 GO SUB 8500: REM COMPRUEBA
4520 IF OK=0 AND PR<60 THEN LET
PR=PR+1: GO TO 4500
4530 RETURN
4998 REM
4999 REM ************************
5000 REM CREA TABLERO
5001 REM ************************
5002 REM
5010 FOR K=1 TO 8
5020 FOR L=1 TO 8
5030 LET T$(K,L)=SCREEN$ (K,L)
5040 NEXT L
5050 NEXT K
5060 RETURN
5065 REM
5070 REM ************************
5075 REM   RECUPERA EL TABLERO
5080 REM ************************
5085 REM
5100 FOR K=1 TO 8: FOR L=1 TO 8
5110 LET T$(K,L)=B$(K,L)
5120 NEXT L: NEXT K
5130 RETURN
5135 REM
5140 REM ************************
5145 REM   GUARDA EL TABLERO
5150 REM ************************
5155 REM
5200 FOR K=1 TO 8: FOR L=1 TO 8
5210 LET B$(K,L)=T$(K,L)
5220 NEXT L: NEXT K
5230 RETURN
6990 REM
6991 REM ************************
6992 REM  EJECUCION DE LA TIRADA
6993 REM ************************
6994 REM
7000 RESTORE 8010
```

```
7010 PRINT AT X,Y;A$
7020 FOR J=1 TO 8
7030 READ IX,IY,FX,FY
7040 IF O(J,1)=0 THEN GO TO 7200
7045 LET X1=X: LET Y1=Y
7050 LET X1=X1+IX: LET Y1=Y1+IY
7060 IF X1=O(J,1) AND Y1=O(J,2)
THEN GO TO 7200
7070 BEEP .1,25: PRINT AT X1,Y1;
A$
7074 IF A$="O" THEN LET CONTO=CO
NTO+1: LET CONTX=CONTX-1
7076 IF A$="X" THEN LET CONTX=CO
NTX+1: LET CONTO=CONTO-1
7100 PRINT AT 12,22;"  ";AT 12,2
2;CONTO
7110 PRINT AT 13,22;"  ";AT 13,2
2;CONTX
7120 GO TO 7050
7200 NEXT J
7300 RETURN
7399 REM
7400 REM ************************
7405 REM     EJECUCION INTERNA
7410 REM ************************
7415 REM
7500 RESTORE 8010
7510 LET T$(X,Y)=A$
7515 LET FG=1
7520 FOR J=1 TO 8
7530 READ IX,IY,FX,FY
7540 IF O(J,1)=0 THEN GO TO 7700
7545 LET X1=X: LET Y1=Y
7550 LET X1=X1+IX: LET Y1=Y1+IY
7560 IF X1=O(J,1) AND Y1=O(J,2)
THEN GO TO 7700
7570 LET T$(X1,Y1)=A$
7580 LET FG=FG+1
7590 GO TO 7550
7700 NEXT J
7800 RETURN
7990 REM
7991 REM ************************
7992 REM COMPROBACION POSIBLIDAD
7993 REM ************************
7994 REM
8000 RESTORE 8010
8010 DATA 1,0,8,0
8011 DATA 0,1,0,8
8012 DATA 0,-1,0,1
8013 DATA -1,0,1,0
8014 DATA 1,1,8,8
8015 DATA -1,-1,1,1
8016 DATA 1,-1,8,1
8017 DATA -1,1,1,8
8020 DIM O(8,2): LET OK=0
8030 FOR J=1 TO 8
8040 READ IX,IY,FX,FY
8050 IF X=FX OR Y=FY THEN GO TO
8200
8055 IF SCREEN$ (X,Y)<>" " THEN
GO TO 8200
8060 LET X1=X+IX: LET Y1=Y+IY
8070 IF SCREEN$ (X1,Y1)=E$ THEN
```

```
GO SUB 8300
8200 NEXT J
8250 RETURN
8300 IF X1=FX OR Y1=FY THEN RETU
RN
8310 LET X1=X1+IX: LET Y1=Y1+IY
8315 IF SCREEN$ (X1,Y1)=" " THEN
 RETURN
8320 IF SCREEN$ (X1,Y1)=A$ THEN
LET OK=1: LET O(J,1)=X1: LET O(J
,2)=Y1: RETURN
8330 GO TO 8300
8399 REM
8400 REM ***********************
8405 REM  COMPROBACION INTERNA
8410 REM ***********************
8415 REM
8500 RESTORE 8010
8510 DIM O(8,2): LET OK=0
8520 FOR J=1 TO 8
8530 READ IX,IY,FX,FY
8540 IF X=FX OR Y=FY THEN GO TO
8600
8550 IF T$(X,Y)<>" " THEN GO TO
8600
8560 LET X1=X+IX: LET Y1=Y+IY
8570 IF T$(X1,Y1)=E$ THEN GO SUB
 8700
8600 NEXT J
8610 RETURN
8700 IF X1=FX OR Y1=FY THEN RETU
RN
8710 LET X1=X1+IX: LET Y1=Y1+IY
8720 IF T$(X1,Y1)=" " THEN RETUR
N
8730 IF T$(X1,Y1)=A$ THEN LET OK
=1: LET O(J,1)=X1: LET O(J,2)=Y1
: RETURN
8750 GO TO 8700
8800 RETURN
8990 REM
8991 REM ***********************
8992 REM      INICIALIZACION
8993 REM ***********************
8994 REM
9000 PRINT AT 0,0;"██████████"
9010 FOR i=1 TO 8: PRINT AT i,0;
"█        █";i: NEXT i
9020 PRINT AT 9,0;"██████████"
9030 PRINT AT 10,1;"12345678"
9040 PRINT AT 4,4;"OX";AT 5,4;"X
O"
9050 LET H$="1881613861386336415
815486564353464572223277426357712
182877272"
9060 LET V$="8811161683833636848
45115435346562775367436257224128
771282772"
9065 LET P$="8888777777776666555
55555444444443333333333333333222
222221111"
9067 DIM B$(8,8): DIM T$(8,8)
9070 LET UJH=1: LET UJO=1
9080 LET CONTO=2: LET CONTX=2
```

```
9090 PRINT AT 12,19;"TU: ";CONTO
9100 PRINT AT 13,19;"YO: ";CONTX
9110 PRINT AT 4,4;"OX"
9120 PRINT AT 5,4;"XO"
9130 PRINT AT 1,17;"INSTRUCCIONE
S";AT 1,17; OVER 1;"____________
"
9140 PRINT AT 3,13;"-TU TIENES L
AS ""O"""
9150 PRINT AT 5,13;"-YO TENGO LA
S ""X"""
9160 PRINT AT 7,13;"-SI NO PUEDE
S TIRAR";AT 8,13;"CONTESTA ""0""
 A ";AT 9,13;"LA JUGADA"
9170 RETURN
9490 REM
9500 REM ***********************
9510 REM   FINAL DE PARTIDA ?
9520 REM ***********************
9525 REM
9540 IF CONTO>CONTX THEN PRINT A
T 17,0;"MUY BIEN, HAS GANADO": B
EEP 10,0: STOP
9550 IF CONTX>CONTO THEN PRINT A
T 17,0;"LASTIMA, HAS PERDIDO": B
EEP 5,-30: STOP
```

## COMENTARIOS SOBRE EL PROGRAMA

En primer lugar observe detenidamente las líneas 100 y 160. En ellas se efectúa la valoración de las jugadas (no de las posiciones), según el siguiente cálculo:

PUNTOS=PUNTOS DE LA POSCION+FICHAS GIRADAS*0.028*(NUMERO DE JUGADA)

Los puntos de la jugada se colocan en la variable P0 si es una jugada de primer nivel (la hace el ordenador) o en la variable PH si es una jugada de segundo nivel (la hace el jugador humano). Los puntos de cada posición están en la variable P$, por eso en la fórmula aparece VAL P$(PR). Donde PR es el número de posición según la tabla de posiciones representada por las variables H$ y V$, que se encuentran especificadas en las líneas 9050 y 9060 de la rutina de inicialización. El número de fichas giradas está contenido en la variable FG que se actualiza en la rutina «EJECUCION INTERNA» (líneas 7015 y 7080). El número de jugada se calcula sumando las fichas de cada bando (CONTO+CONTX), variables que se inicializan en la rutina de inicialización y que se van actualizando en la rutina «EJECUCION DE LA TIRADA» (líneas 7074 y 7076). Por último tenemos el coeficiente 0.028 que modifica el valor del producto entre las fichas giradas y el número de jugada.
El segundo término de la fórmula (FG*0.028*(CONTO+CONTX) hace que entre dos jugadas que giren el mismo número de fichas, tenga

más valor la que esté más al final de la partida. Si el coeficiente fuera 0 el número de fichas giradas no tendría valor. Y si fuera un número grande (1, por ejemplo) la consecuencia sería que a partir de un momento de la partida (jugada 5 o 6) la posición perdería su importancia mientras que el número de fichas giradas iría adquiriendo más cada vez. Por todo esto el número 0.028 se puede cambiar según la estrategia con la que se quiere que juegue la máquina. Este número lo he elegido por tanteo ya que me ha parecido que era el que hacía que el ordenador jugara mejor, pero usted puede tener otra opinión.
Obsérvese que debido a la línea 105 que inicializa MIN con un valor muy grande (10000) inalcanzable por cualquier puntuación, la línea 210 coloca en ella el valor mínimo (el peor para el ordenador) de todas las respuestas a una jugada de primer nivel. Por tanto MIN corresponde a las posiciones de tipo PX donde en este caso, X puede valer desde 1 hasta 60. Observe también que gracias a esto no hace falta guardarse todas las puntuaciones de segundo nivel (P0-PH, en el programa) sino que se guarda únicamente la mínima en la variable MIN.
En la línea 300 se elige una jugada de primer nivel, en el caso de que suponiendo que el contrario nos responda con su mejor respuesta (donde se obtendría la puntuación que hay en ese momento en MIN), ésta sea de mejor puntuación que la anteriormente elegida, es decir, que MIN sea mayor que MAX.
La línea 200 es muy importante, ya que hace que si en una rama se ha obtenido una puntuación peor que la mejor hasta el momento (almacenada en MAX), ya no se examine el resto de la rama y se pase por tanto a examinar la siguiente jugada de primer nivel. Por último, si desea introducir este programa en el ordenador para competir con él, léase antes los consejos del último capítulo, ya que conseguirá un ahorro considerable en el tiempo de respuesta.

# JUEGOS CON TRUCO: EL 15

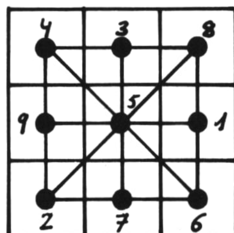
El juego que vamos a ver ahora está sacado de una idea de Martin Gardner. El juego consiste en lo siguiente: Se tiene una hilera de casillas que contienen los número del 1 al 9. Los jugadores van eligiendo números por turnos, de modo que si un número ha sido ya elegido no puede serlo otra vez por ninguno de los jugadores. Se declara vencedor, al jugador de los que consiga tener tres números distintos que sumen 15. La idea es sencilla, ¿verdad? Pero ¿cómo lo haremos para hacer un programa que juegue al 15 de un modo eficaz? Una manera sería decirle a la máquina cuáles son las combinaciones ganadoras y utilizando el método del árbol analizar las jugadas puntuándolas más si «prometen» más, o puntuarlas mucho si consiguen una combinación ganadora.
Pues bien la idea está en que este juego es totalmente equivalente al tres en raya, que es un juego muy fácil de programar y del cual existen ya muchos programas.
Vamos a analizar cuáles son las combinaciones ganadoras:

```
1 6 8        3 5 7
2 5 8        3 4 8
1 5 9        2 6 7
4 5 6        2 4 9
```

Todas esas ternas de números tienen la particularidad de que suman 15, y además son las únicas que tienen esta propiedad.
Imaginemos ahora el siguiente cuadrado:

| | | |
|---|---|---|
| 4 | 3 | 8 |
| 9 | 5 | 1 |
| 2 | 7 | 6 |

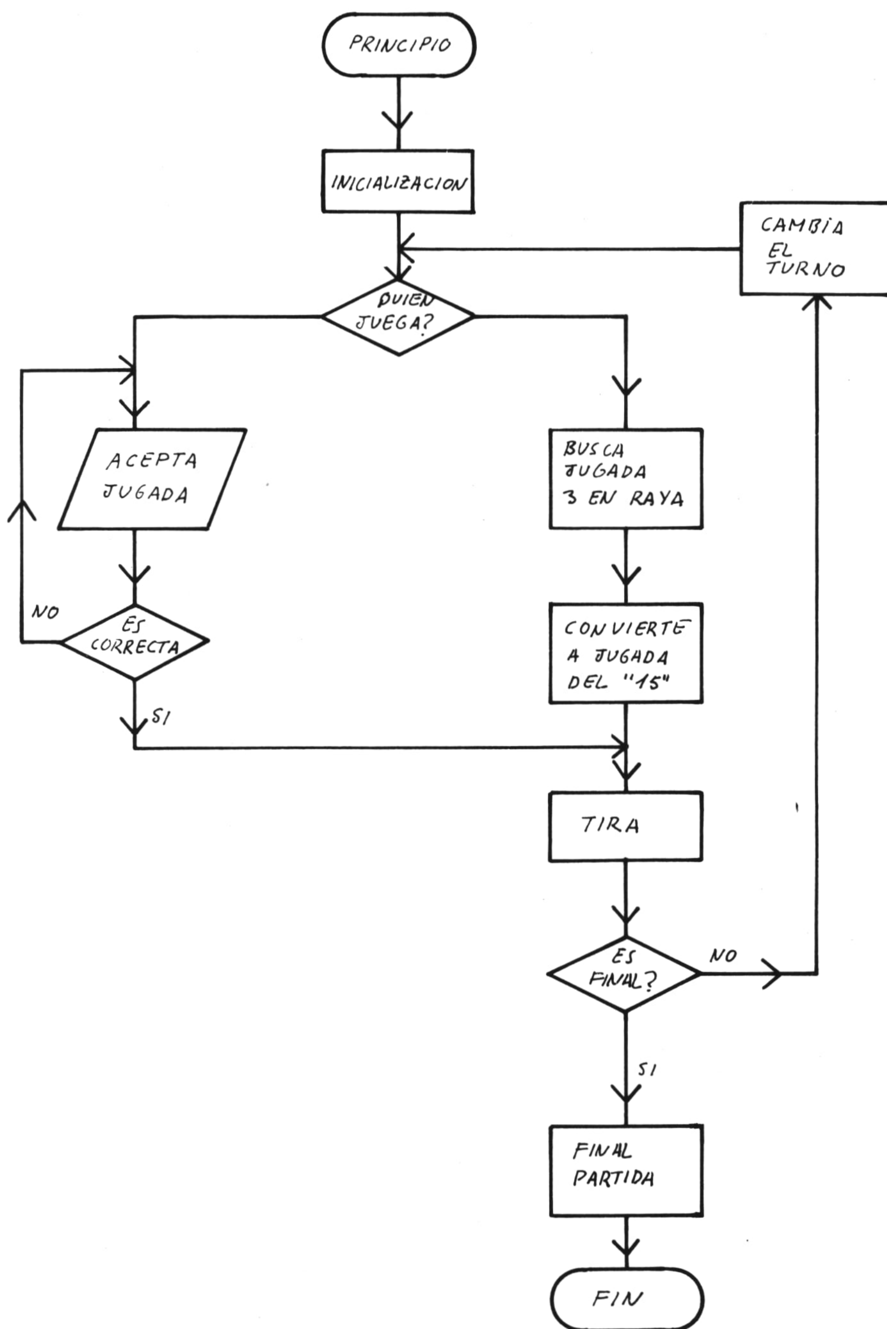
Si lo observamos con detenimiento, veremos que las tres columnas, las tres filas y las dos diagonales, forman 8 líneas rectas que contienen a las únicas 8 combinaciones ganadoras del 15:




Por lo tanto, jugar al 15 es lo mismo que jugar al 3 en raya, teniendo la precaución de cambiar la casilla por el número que está contenido en ella.
Para hacer una programa que juegue al tres en raya, normalmente no se usa la búsqueda por árbol, ya que con unas pocas reglas, se consigue al menos hacer tablas. Para los que quieran desarrollarlo por el método tradicional escribiremos la parte de la jugada del ordenador en pseudolenguaje, ya que en el libro lo desarrollaremos por árbol a tres niveles, más que nada para ejemplificar lo que se acaba de explicar:

```
—JUGADA DEL ORDENADOR—
SI CONSIGUES 3 EN RAYA TIRA EN ESA CASILLA
SI EVITAS UN TRES EN RAYA, TIRA EN ESA CASILLA
SI CONSIGUES UN DOS EN RAYA, TIRA EN ESA CASILLA
SI EVITAS UN DOS EN RAYA, TIRA EN ESA CASILLA
SI HAY UNA RAYA VACIA, TIRA EN CUALQUIER CASILLA QUE
LE PERTENEZCA
SI NO OCURRE NADA DE LO ANTERIOR. TIRA EN LA PRIMERA
CASILLA VACIA QUE ENCUENTRES.
—FIN—
```

Ahora vamos a complicarlo para desarrollarlo para un árbol de tres niveles. El programa no tardaría mucho más si lo hacemos hasta el último nivel, ya que la primera jugada siempre será la casilla del medio (correspondiente al número 5 del 15) en caso de que esté desocupada. Así por ejemplo, en caso de que empieze el contrario, y tire en una casilla distinta de la 5, el ordenador contestará tirando en la 5 inmediatamente y el jugador a su vez responderá tirando en una casilla cualquiera. Por lo tanto, sólo quedarán 6 casillas por llenar, de modo que aproximadamente, el número total de combinaciones de juego que quedan es de 6×5×4×3×2×1=720. Si en cambio empieza el ordenador tirando en la casilla 5, después de la jugada del contrario habrá 7×6×5×4×3×2×1 posibilidades, es decir, 5040 combinaciones. El caso peor es cuando empieza el jugador y además tira en la casilla 5 ya que el número de combinaciones a examinar sería de más de 40000. Creo que esto es suficiente como para reducir el nivel a 3 ya que con ello, el caso peor queda reducido a 336 posibilidades, y el mejor a 120.
El diagrama es parecido al de los demás juegos:


PRINCIPIO
INICIALIZACION
CAMBIA EL TURNO
QUIEN JUEGA?
ACEPTA JUGADA
NO
ES CORRECTA
SI
BUSCA JUGADA 3 EN RAYA
CONVIERTE A JUGADA DEL "15"
TIRA
ES FINAL?
NO
SI
FINAL PARTIDA
FIN

Prácticamente la única diferencia que hay es que la jugada del ordenador (que la ha pensado como 3 en raya) debe convertirse en una jugada del 15 antes de tirar. Esto no ocurre con la jugada del humano ya que se supone que no conoce el truco, y en caso de que lo conozca ¡al menos que pierda un tiempo pensando!
Por esta vez no escribiré el programa en pseudolenguaje ya que es similar al del Othello, sólo que con un nivel más, así que vamos a por el listado:

```
  10 GO SUB 8000: REM INICIALIZA
CION
  20 INPUT "EMPIEZO YO? ";R$
  25 IF R$<>"NO" AND R$<>"SI" TH
EN GO TO 20
  30 IF R$="NO" THEN GO TO 1200
  40 REM
  50 REM ***********************
  60 REM PRINCIPIO PRIMER NIVEL
  70 REM ***********************
  80 REM
  90 LET PUN1=-10000
  95 IF CONT<2 AND A(2,2)=0 THEN
 LET X=2: LET Y=2: LET JUG=1: GO
 TO 1050
 100 LET P1=1
 110 IF P1>9 THEN GO TO 1040
 120 LET X1=C(P1,1): LET Y1=C(P1
,2)
 130 IF A(X1,Y1)<>0 THEN LET P1=
P1+1: GO TO 110
 135 GO SUB 3000: REM CAMBIA TAB
LERO
 300 LET T(X1,Y1)=1: REM TIRADA
 310 GO SUB 6000
 320 GO SUB 5000
 330 LET FINAL=0: IF F$(1)="L" T
HEN LET JUG=P1: LET X=X1: LET Y=
Y1: GO TO 1040
 350 REM
 360 REM ***********************
 370 REM      SEGUNDO NIVEL
 380 REM ***********************
 390 REM
 400 LET PUN2=10000
 410 LET P2=1
 420 IF P2>9 THEN GO TO 1015
 430 LET X2=C(P2,1): LET Y2=C(P2
,2)
 440 IF T(X2,Y2)<>0 THEN LET P2=
P2+1: GO TO 420
 490 REM ***********************
 500 REM
 510 REM PRINCIPIO TERCER NIVEL
 520 REM
 530 REM ***********************
 600 LET T(X2,Y2)=4
 700 LET PUN3=-10000
 710 LET P3=1
```

```
 720 IF P3>9 THEN GO TO 1005
 730 LET X3=C(P3,1): LET Y3=C(P3
,2)
 740 IF T(X3,Y3)<>0 THEN LET P3=
P3+1: GO TO 720
 750 LET T(X3,Y3)=1
 755 PRINT AT 1,7; OVER 1;"PIENS
O....."
 760 GO SUB 6000: REM LEE FILAS
 770 GO SUB 6063: REM PUNTUA
1000 IF PUN>PUN3 THEN LET PUN3=P
UN:
1004 IF P3<9 THEN LET P3=P3+1  L
ET T(X3,Y3)=0: GO TO 720
1005 REM _______________________
1007 REM    FIN TERCER NIVEL
1008 REM _______________________
1009 REM
1010 IF PUN3<PUN2 THEN LET PUN2=
PUN3
1011 IF PUN2<=PUN1 THEN GO TO 10
30
1015 IF P2<9 THEN LET P2=P2+1: L
ET T(X3,Y3)=0: LET T(X2,Y2)=0: G
O TO 420
1020 REM _______________________
1021 REM   FIN SEGUNDO NIVEL
1022 REM _______________________
1023 REM
1025 IF PUN2>PUN1 THEN LET PUN1=
PUN2: LET X=X1: LET Y=Y1: LET JU
G=P1
1030 IF P1<9 THEN LET P1=P1+1: L
ET T(X3,Y3)=0: LET T(X2,Y2)=0: G
O TO 110
1031 REM _______________________
1033 REM     FIN PRIMER NIVEL
1040 REM _______________________
1045 REM
1060 LET A(X,Y)=1: LET CONT=CONT
+1
1070 BEEP 0.1,0: PRINT AT 10,3*V
AL J$(JUG); INVERSE 1;VAL J$(JUG
)
1075 GO SUB 3000: REM CAMBIA TAB
LERO
1080 GO SUB 6000: REM LEE TABLER
O
1090 GO SUB 5000: REM MIRA SI HA
Y 3 EN RAYA
1100 IF FINAL=1 THEN GO TO 9900
1110 IF CONT=9 THEN GO TO 9900
1199 REM
1200 REM ***********************
1205 REM TIRADA DEL JUGADOR
1210 REM ***********************
1215 REM
1230 INPUT "QUE NUMERO QUIERES?
";N
1240 IF N<>INT N OR N>9 OR N<1 T
HEN GO TO 1230
1250 LET T=VAL K$(N)
1260 LET X=C(T,1): LET Y=C(T,2)
1270 IF A(X,Y)<>0 THEN GO TO 123
0
```

```
1280 LET A(X,Y)=4: LET JUG=T: LE
T CONT=CONT+1
1285 BEEP 0.1,25: PRINT AT 10,3*
N; BRIGHT 1;N
1290 GO SUB 3000: REM CAMBIA TAB
LERO
1300 GO SUB 6000: REM LEE TABLER
O
1310 GO SUB 5000: REM  MIRA 3 EN
 RAYA
1400 IF FINAL=1 THEN GO TO 9900
1410 IF CONT=9 THEN GO TO 9900
1420 GO TO 50
2999 REM
3000 REM ***********************
3010 REM  TRASPASO DEL TABLERO
3020 REM ***********************
3030 REM
3040 FOR I=1 TO 3
3050 FOR J=1 TO 3
3060 LET T(I,J)=A(I,J)
3080 NEXT J: NEXT I
3090 RETURN
5000 REM
5001 REM ***********************
5002 REM  MIRA SI HAY 3 EN RAYA
5005 REM ***********************
5006 REM
5010 FOR K=1 TO 8
5020 IF L(K)=12 THEN LET FINAL=1
: LET F$="MUY BIEN, HAS GANADO"
5030 IF L(K)=3 THEN LET FINAL=1:
 LET F$="LO SIENTO, HE GANADO"
5040 NEXT K
5050 RETURN
5890 REM
5900 REM ***********************
5905 REM     LEE LAS FILAS
5910 REM ***********************
5920 REM
6000 DIM L(8)
6004 FOR I=1 TO 3
6010 FOR J=1 TO 3
6020 LET L(I)=L(I)+T(I,J): LET L
(3+I)=L(3+I)+T(J,I)
6030 NEXT J: NEXT I
6040 FOR I=1 TO 3
6050 LET L(7)=L(7)+T(I,I): LET L
(8)=L(8)+T(I,4-I)
6055 NEXT I
6058 RETURN
6060 REM
6062 REM ***********************
6063 REM         PUNTUA
6064 REM ***********************
6065 REM
6066 LET PUN=0
6070 FOR L=1 TO 8
6080 LET PUN=PUN+(10 AND L(L)=1)
+(50 AND L(L)=2)+(150 AND L(L)=3
)-(10 AND L(L)=4)-(50 AND L(L)=8
)-(5000 AND L(L)=12)
6090 NEXT L
6100 RETURN
```

```
7989 REM
7990 REM ***********************
7991 REM      INICIALIZACION
7995 REM ***********************
7996 REM
8000 DIM L(8): DIM C(9,2): DIM A
(3,3): DIM T(3,3)
8001 LET CONT=0
8100 LET J$="524689731"
8200 LET K$="928314756"
8300 CLS
8310 PRINT AT 9,2;"▛▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀"
8320 PRINT AT 10,2;"▌1 ▌2 ▌3 ▌4 ▌5 ▌6 ▌7 ▌8 ▌9 ▌"
8330 PRINT AT 11,2;"▙▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄"
8400 LET F$="TABLAS"
8500 LET FINAL=0
8600 LET CONT=0
9000 DATA 2,2,1,1,1,3,3,1,3,3,1,
2,2,1,2,3,3,2
9010 RESTORE 9000
9020 FOR I=1 TO 9
9030 READ C(I,1),C(I,2)
9040 NEXT I
9050 RETURN
9799 REM
9800 REM ***********************
9805 REM     FINAL DE PARTIDA
9810 REM ***********************
9820 REM
9900 BEEP 3,20
9910 PRINT AT 16,5;F$
9920 INPUT "OTRA PARTIDA? ";R$
9930 IF R$<>"SI" AND R$<>"NO" TH
EN GO TO 9920
9940 IF R$="SI" THEN RUN
9950 PRINT "COBARDE!!!!"
```

## VARIABLES UTILIZADAS

— A(3,3). Representa el tablero interno real (sobre el que se juega) de 3 en raya.
— T(3,3). Es el tablero interno de 3 en raya utilizado para hacer las valoraciones.
— C(9,2) Al igual que en el Othello nos proporciona ordenadamente las posibles jugadas en orden de mejor a peor, para poder beneficiarnos ventajosamente del posible desecho de una rama del árbol. El orden es bastante lógico: En primer lugar la casilla del centro, ya que pertenece a 4 líneas. En segundo lugar las de las esquinas que pertenecen a 2 líneas, y en último lugar las casillas que ocupan el centro de los lados del cuadrado, ya que sólo pertenecen a una de las líneas.
— L(8). Almacena la suma de valores de cada línea. En cada casilla,

se pone un 4 si hay una ficha del jugador humano, y se pone un 1 si contiene una ficha del ordenador.

— PUN. Contiene los puntos de la posición después de una jugada de último nivel.

— PUN1, PUN2, PUN3. Contienen las puntuaciones de cada nivel (PUN1 es el máximo de los distintos PUN2, que a su vez es el mínimo de los distintos PUN3, que es el máximo de los PUN).

— P1, P2, P3. Contienen el número de jugada que se está analizando en los niveles respectivos según el orden especificado por la variable C(9,2).

— X1,Y1,X2,Y2,X3,Y3, Son las coordenadas en el tablero de P1, P2 y P3 respectivamente.

— X,Y. Coordenadas de la jugada seleccionada.

— JUG. Número de la jugada seleccionada según el orden especificado por C(9,2).

— N. Número (del tablero del 15) en el que el jugador deposita su ficha.

— K$. Datos para convertir una casilla del 15 en una casilla del 3 en raya. Siempre se apoya en el orden de C(9,2).

— J$. Al revés que el anterior, se usa para convertir una casilla del 3 en raya en una casilla del 15.

— T. Es el número N ya convertido en una casilla del 3 en raya.

— CONT. Contador que indica el número de fichas jugadas.

Con estas especificaciones no le será difícil seguir el flujo del programa. La rutina de la jugada del ordenador se compone de 3 grandes bucles anidados, es decir, uno dentro del otro. El más interior, que es el que examina las jugadas de tercer nivel, sólo termina cuando se han examinado todas, ya que no sabemos si la próxima respuesta puede tener mejor puntuación aunque las demás hayan sido malas. Lo mismo ocurre con el de primer nivel, exceptuando el caso de que se consiga un 3 en raya (líneas 310 a 330), o bien en el caso de que no esté ocupada la casilla 5 (línea 95). En cambio el bucle de segundo nivel termina en caso de que para una de esas jugadas se obtenga como mejor resultado uno peor que el de la jugada que se ha escogido como mejor hasta el momento (línea 1025).

La puntuación de una jugada se obtiene en la subrutina «PUNTUA» de la línea 6065. Esto se hace después de haber pasado por la subrutina «LEE TABLERO» de la línea 6000. Si indicamos las fichas de ordenador por Os y las del jugador por Hs tenemos la siguiente relación entre la lectura y la puntuación:

| LEIDO | SIGNIFICADO | PUNTOS |
|---|---|---|
| 1 | 1 O | 10 |
| 2 | 2 Os | 50 |
| 3 | 3 Os | 150 |
| 4 | 1 H | —10 |
| 5 | 1 O y 1 H | 0 |
| 6 | 2 Os y 1 H | 0 |
| 8 | 2 Hs | —50 |
| 9 | 1 O y 2 Hs | 0 |
| 12 | 3 Hs | —5000 |

Las lecturas 7, 10 y 11 no se pueden obtener, y las lecturas 9, 6 y 5 valen 0 porque se trata de líneas que no se pueden ganar ni perder. La puntuación de —5000 para la lectura 12 es totalmente simbólica Aquí también se puede variar la estrategia, cambiando el método de puntuar. Así, poniendo más puntos a las jugadas negativas que a sus equivalentes positivas, conseguiremos una estrategia defensiva, mientras que haciendo lo contrario conseguiremos una estrategia atacante.

# JUEGOS GRAFICOS

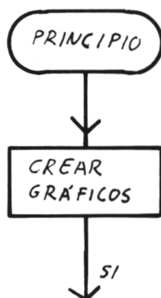
«Juegos gráficos» es un concepto muy general que engloba a muchos tipos de juegos con una característica común: Hacen un uso continuado de las posibilidades gráficas del ordenador. Entre ellos tenemos a los juegos de pelota, los juegos del tipo «arcade» o «marcianitos», los juegos de circuito, etc. En esta sección desarrollaremos una versión de un programa de un juego de circuito muy conocido: PAC-MAN o FANTASMAS.

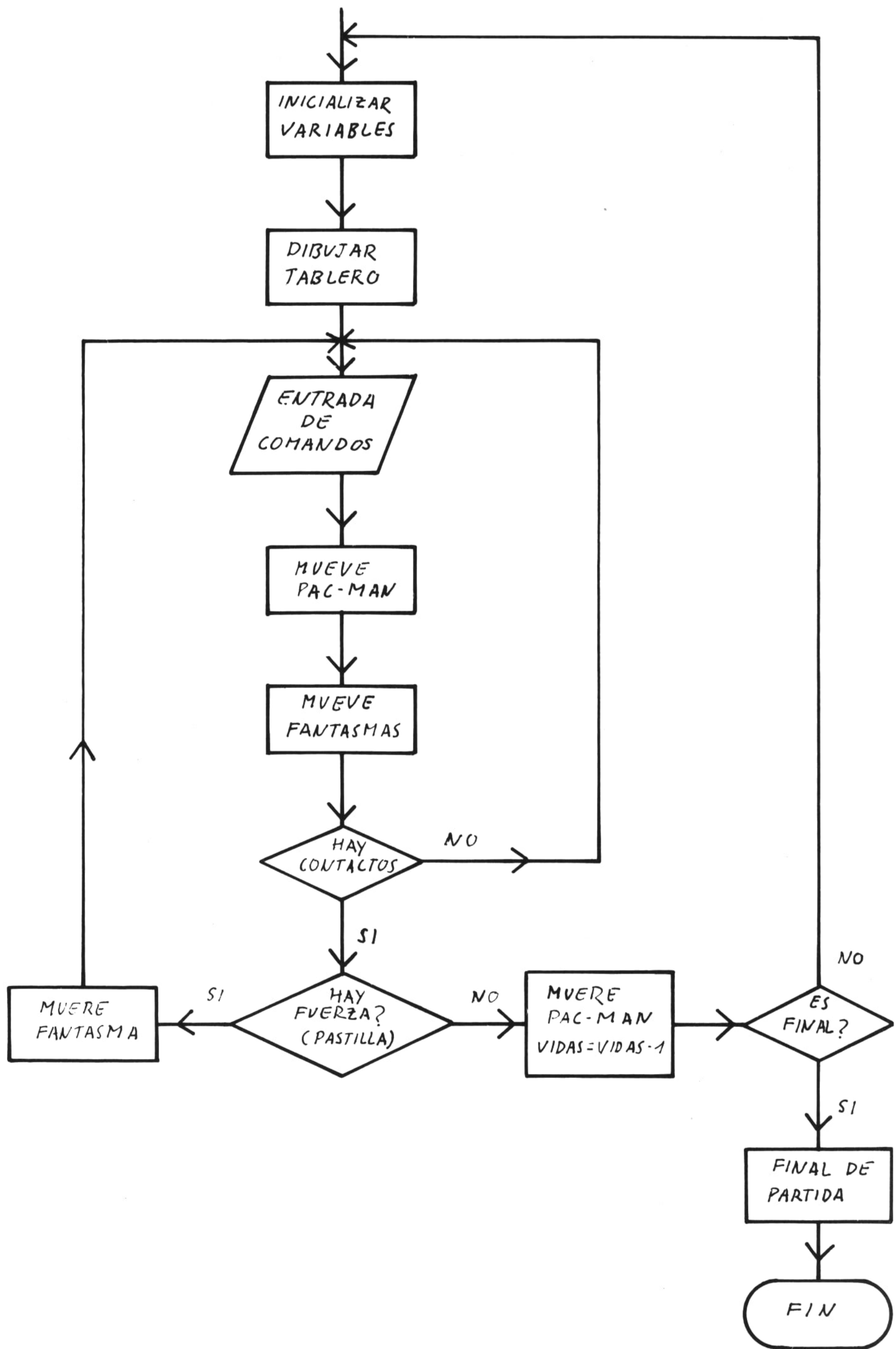
En este tipo de juegos, como ocurre con los marcianitos y con los de pelota, es muy importante la rapidez, ya que al ser interactivos, si el programa no es lo suficientemente rápido, nos dará la sensación de que no responde a nuestras «insistentes» órdenes. Lo que en realidad ocurre es que el programa aún no ha llegado a la línea o líneas que tiene destinadas a aceptar nuestras entradas, ya sean desde teclado, o desde un mando de juegos.

Esta es la razón de que el juego no tenga más que fantasmas que persiguen al PAC-MAN.

En realidad, el programa principal es muy corto, pues consta nada más que de un bucle en el que se aceptan los comandos del teclado, según sean estos comandos se transmite un movimiento u otro al PAC-MAN y se realiza un movimiento automático de los fantasmas de modo que le persigan. Aparte de esto, que es lo principal, hay que prever el caso en que el PAC-MAN se come las «pastillas» de modo que cuando tenga un encuentro el fantasma no le reste una vida, y en lugar de «morirse» el PAC-MAN se muera el FANTASMA.

El resto del programa, consistirá en dibujar el tablero, inicializar las variables y crear los gráficos de todos los objetos que en él aparecen.




INICIALIZAR VARIABLES
DIBUJAR TABLERO
ENTRADA DE COMANDOS
MUEVE PAC-MAN
MUEVE FANTASMAS
HAY CONTACTOS
NO
SI
HAY FUERZA? (PASTILLA)
SI
MUERE FANTASMA
NO
MUERE PAC-MAN VIDAS=VIDAS-1
ES FINAL?
NO
SI
FINAL DE PARTIDA
FIN

```
  1 REM
  2 REM ***********************
  4 REM      INICIALIZACION
  5 REM ***********************
  6 REM
 10 RESTORE : LET record=0: INK 7: PAPER 1: BORDER 1: CLS : GO SUB 9000
 20 LET tpun=383: LET tin=2: LET pt=0
 30 LET p$="●"
 40 LET v=3
 50 LET punt=0: LET x=11: LET y=15
 60 LET a=2: LET b=2
 70 LET e=19: LET f=29
 99 REM
100 REM ***********************
105 REM    CREACION TABLERO
110 REM ***********************
115 REM
220 DIM a$(21,32)
240 LET a$(1)="███████████████████████████████"
250 LET a$(2)="█*............██............*██"
260 LET a$(3)="█.████.██████.██.██████.████.██"
270 LET a$(4)="█ная............................██"
280 LET a$(5)="█.████.█.████████████.█.████.██"
290 LET a$(6)="█......█......██......█......██"
300 LET a$(7)="████.█.██████.██.██████.█.█████"
310 LET a$(8)="████.█.█..............█.█.█████"
320 LET a$(9)="████.█.█.████████████.█.█.█████"
330 LET a$(10)=".........████████████.........█"
340 LET a$(11)="████.█.█.████████████.█.█.█████"
370 LET a$(12)="████.█.█..............█.█.█████"
380 LET a$(13)="████.█.█.████████████.█.█.█████"
390 LET a$(14)="████.█.█......██......█.█.█████"
400 LET a$(15)="█........████.██.████........██"
410 LET a$(16)="█.████.█..............█.████.██"
420 LET a$(17)="█...██.█.████████████.█.██...██"
430 LET a$(18)="█.█....█......██......█....█.██"
440 LET a$(19)="█.█.█████████.██.█████████.█.██"
445 LET a$(20)="█............................██"
500 PRINT AT 21,0;"████████████████████████████████"
```

```
 600 FOR z=1 TO 20: PRINT AT z,0
;"■";a$(z): NEXT z
 900 PRINT AT 0,10;"VIDAS=";v
 910 PRINT AT 0,0;"PUNTOS=";punt
 990 PAUSE 0
 991 REM
 992 REM ************************
 993 REM  INICIO BUCLE PRINCIPAL
 994 REM ************************
 995 REM
1000 LET a$(x,y)=" "
1001 PRINT AT x,y; INK 7;p$
1002 PRINT AT e,f; INK tin;"ᗣ"
1003 PRINT AT a,b; INK tin;"ᗣ"
1004 PRINT AT 0,6; INK 7;punt;AT
 0,16;v
1005 LET pt=pt-1
1007 IF INKEY$="" THEN GO TO 104
5
1010 IF INKEY$="a" THEN LET p$="
ᗢ"
1020 IF INKEY$="z" THEN LET p$="
ᗜ"
1030 IF INKEY$="l" THEN LET p$="
◖"
1040 IF INKEY$="k" THEN LET p$="
◗"
1060 IF p$="◗" AND a$(x,y-1)<>"■
" THEN PRINT AT x,y;" ": LET y=y
-1
1070 IF p$="ᗢ" AND a$(x-1,y)<>"■
" THEN PRINT AT x,y;" ": LET x=x
-1
1080 IF p$="ᗜ" AND a$(x+1,y)<>"■
" THEN PRINT AT x,y;" ": LET x=x
+1
1090 IF p$="◖" AND a$(x,y+1)<>"■
" THEN PRINT AT x,y;" ": LET y=y
+1
1130 IF a$(x,y)="*" THEN LET pun
t=punt+10: LET pt=20: FOR z=0 TO
 5: BEEP .005,z*5: NEXT z: GO TO
 1160
1140 IF a$(x,y)="." THEN LET pun
t=punt+1: BEEP .001,50
1160 PRINT AT x,y; INK 7;"●": BE
EP .001,30
1170 IF pt>0 THEN LET tin=4
1180 IF pt<0 THEN LET tin=2
1200 IF y>29 THEN PRINT AT x,y;"
 ": LET y=2
1210 IF y<2 THEN PRINT AT x,y;"
": LET y=29
1230 LET p1=INT (RND*7)
1240 LET p2=INT (RND*7)
1245 PRINT AT a,b; INK 7;a$(a,b)
;AT e,f; INK 7;a$(e,f)
1250 IF p2>5 THEN GO TO 1260
1251 IF a>x AND a$(a-1,b)<>"■" T
HEN LET a=a-1
1252 IF b<y AND a$(a,b+1)<>"■" T
HEN LET b=b+1
1253 IF b>y AND a$(a,b-1)<>"■" T
HEN LET b=b-1
```

```
1254 IF x>a AND a$(a+1,b)<>"█" T
HEN LET a=a+1
1260 IF p1>5 THEN GO TO 1270
1261 IF e<x AND a$(e+1,f)<>"█" T
HEN LET e=e+1
1262 IF f<y AND a$(e,f+1)<>"█" T
HEN LET f=f+1
1263 IF f>y AND a$(e,f-1)<>"█" T
HEN LET f=f-1
1264 IF e>x AND a$(e-1,f)<>"█" T
HEN LET e=e-1
1300 IF pt<0 THEN GO SUB 3000
1310 IF pt<0 THEN GO TO 1340
1320 IF (x=e AND y=f) THEN FOR m
=0 TO 5: FOR z=20 TO 30: BEEP .0
05,z: NEXT z: NEXT m: PRINT AT e
,f;a$(e,f): LET e=19: LET f=29
1330 IF (a=x AND b=y) THEN FOR m
=0 TO 3: FOR z=20 TO 30: BEEP .0
05,z: NEXT z: NEXT m: PRINT AT a
,b;a$(a,b): LET a=2: LET b=2
1900 IF punt>=383 THEN GO TO 500
0
1980 IF v<=0 THEN GO TO 2000
1999 GO TO 1000
2100 FOR m=0 TO 2: FOR z=-10 TO
50: BEEP .005,z: NEXT z: NEXT m:
 PRINT AT 11,8;"FIN DE PARTIDA":
 IF punt>record THEN LET record=
punt
2110 PRINT AT 13,6; FLASH 1; INK
 1; PAPER 6;"El record es:";reco
rd
2120 INPUT "PULSA ENTER PARA JUG
AR"; LINE q$: GO TO 2
3000 IF (a=x AND b=y) OR (e=x AN
D f=y) THEN PRINT AT x,y;" ": FO
R m=0 TO 4: FOR z=10 TO 20: BEEP
 .01,-z: NEXT z: NEXT m: LET x=1
1: LET y=14: LET r$="●": LET v=v
-1
3100 RETURN
5000 FOR z=0 TO 5: FOR m=-10 TO
10: BEEP .01,m: NEXT m: NEXT z:
LET tpunt=tpunt+383
5005 LET x=11: LET y=15
5010 GO TO 100
8995 REM
8996 REM ************************
8997 REM   CREACION CARACTERES
8998 REM ************************
8999 REM
9000 DATA 0
9001 DATA BIN 01000010
9002 DATA BIN 11100111
9003 DATA BIN 11111111
9004 DATA BIN 11011011
9005 DATA BIN 11111111
9006 DATA BIN 01111110
9007 DATA BIN 00111100
9008 REM
9010 DATA BIN 00111100
9011 DATA BIN 01111110
9012 DATA BIN 11101100
```

```
9013 DATA BIN 11111000
9014 DATA BIN 11111000
9015 DATA BIN 11101100
9016 DATA BIN 01111110
9017 DATA BIN 00111100
9018 REM
9020 DATA BIN 00111100
9021 DATA BIN 01111110
9022 DATA BIN 00110111
9023 DATA BIN 00011111
9024 DATA BIN 00011111
9025 DATA BIN 00110111
9026 DATA BIN 01111110
9027 DATA BIN 00111100
9028 REM
9030 DATA BIN 00111100
9031 DATA BIN 01111110
9032 DATA BIN 11111111
9033 DATA BIN 11011011
9034 DATA BIN 11111111
9035 DATA BIN 11100111
9036 DATA BIN 01000010
9037 DATA BIN 00000000
9038 REM
9040 DATA BIN 00111100
9041 DATA BIN 01111110
9042 DATA BIN 11111111
9043 DATA BIN 11011011
9044 DATA BIN 11111111
9045 DATA BIN 10100101
9046 DATA BIN 10100101
9047 DATA BIN 10100101
9048 REM
9050 DATA BIN 00111100
9051 DATA BIN 01111110
9052 DATA BIN 11111111
9053 DATA BIN 11111111
9054 DATA BIN 11111111
9055 DATA BIN 11111111
9056 DATA BIN 01111110
9057 DATA BIN 00111100
9060 FOR f=0 TO 7: READ a: POKE
USR "[graphic]"+f,a: NEXT f
9065 FOR f=0 TO 7: READ a: POKE
USR "[graphic]"+f,a: NEXT f
9070 FOR f=0 TO 7: READ a: POKE
USR "[graphic]"+f,a: NEXT f
9075 FOR f=0 TO 7: READ a: POKE
USR "[graphic]"+f,a: NEXT f
9080 FOR f=0 TO 7: READ a: POKE
USR "[graphic]"+f,a: NEXT f
9090 FOR f=0 TO 7: READ a: POKE
USR "[graphic]"+f,a: NEXT f
9900 RETURN
```

## VARIABLES USADAS

— a$(21,32). Almacena el tablero.
— t. Color de la tinta del fantasma.
— pas. Es un «switch» para saber si el PAC-MAN se ha comido una

pastilla. También cuenta el tiempo que ha pasado desde que se la comió.

— p$. Carácter del PAC-MAN en un momento determinado.
— x,y. Coordenadas del PAC-MAN.
— f1,g1. Coordenadas del primer fantasma.
— f2,g2.— Coordenadas del segundo fantasma.
— récord. Puntuación más alta hasta el momento.
— sx. Determina si el fantasma X perseguirá o no al PAC-MAN en esta iteración del bucle.

# GENERALIDADES

## PRESENTACION DE LOS PROGRAMAS

Al programa de fantasmas, poco más se le puede añadir en cuanto a la presentación. No ocurre lo mismo con los programas de juegos inteligentes, que se han presentado completamente «desnudos» para que usted los pueda «vestir» a su gusto. Ahí van algunas ideas:
Para el programa del NIM se podría crear algún carácter gráfico que representara al objeto que se retira. De modo que en lugar de salir por pantalla el texto indicando el número de objetos que se retiran, saliera un «monstruito» que se los fuera llevando. Para hacer esto quizás necesitara limitar el número de objetos entre dos cantidades precisas.
En lo que se refiere a los programas del Othello, aparte de que se les puede añadir colorido, el tablero podría ser el doble de grande, para ello sólo necesitará modificar las instrucciones en que aparezcan las sentencias PRINT o SCREEN$, multiplicando por 2 las coordenadas. Otra cosa que se podría hacer es crear un gráfico para cada bando (que puede estar formado por 4 si hace el tablero doble). Se podrían hacer muchas cosas más, pero considero que se debe evitar a toda costa redibujar el tablero cada vez que se cambia o se añade una ficha ya que entonces no se recuerda bien la posición antigua.
El siguiente programa nos permite realizar una buena presentación en el juego del Othello. Para ello lo haremos lo más parecido a la realidad. En él, cada ficha se divide en cuatro partes, la superior izquierda, la superior derecha, la inferior izquierda y la inferior derecha. Como se puede observar en el listado, en cada parte se definen también las líneas divisorias del tablero en el caso de que a esa parte le corresponda estar ya sea en un cuadrado inferior o en el lado izquierdo de la ficha.
Se han definido 8 caracteres, de los cuales 4 pertenecen a las fichas blancas y 4 a las negras. Tenga en cuenta que en el listado sale el gráfico y que para que esto ocurra tiene que pulsar SHIFT-G y luego la letra correspondiente habiendo ejecutado previamente el programa hasta la línea 1360, ya que en caso contrario le aparecerá el carácter y no el gráfico. Para ello consulte la distribución de los gráficos que se da más adelante.

```
   1 REM ***********************
   2 REM    DIBUJO DEL TABLERO
   3 REM ***********************
  10 PAPER 4
  20 BORDER 4
  30 FOR i=24 TO 176-24 STEP 16
  40 PLOT 64,i: DRAW 128,0
  50 NEXT i
  60 FOR i=64 TO 256-64 STEP 16
  70 PLOT i,24: DRAW 0,128
  80 NEXT i
  90 PRINT AT 2,7;"█████████████
█████"
 100 PRINT AT 19,7;"████████████
██████"
 110 FOR i=3 TO 18
 120 PRINT AT i,7;"█";AT i,24;"█
"
 130 NEXT i
 200 REM ***********************
 204 REM  SUP. IZQ. FICHA BLANCA
 205 REM ***********************
 210 DATA BIN 10000000
 220 DATA BIN 10000011
 230 DATA BIN 10001110
 240 DATA BIN 10011000
 250 DATA BIN 10110000
 260 DATA BIN 10110000
 270 DATA BIN 11100000
 280 DATA BIN 11100000
 300 REM ***********************
 303 REM  SUP. DER. FICHA BLANCA
 305 REM ***********************
 310 DATA BIN 00000000
 320 DATA BIN 11000000
 330 DATA BIN 01110000
 340 DATA BIN 00011000
 350 DATA BIN 00001100
 360 DATA BIN 00001100
 370 DATA BIN 00000110
 380 DATA BIN 00000110
 400 REM ***********************
 402 REM  INF. IZQ. FICHA BLANCA
 405 REM ***********************
 410 DATA BIN 11100000
 420 DATA BIN 11100000
 430 DATA BIN 10110000
 440 DATA BIN 10110000
 450 DATA BIN 10011000
 460 DATA BIN 10001110
 470 DATA BIN 10000011
 480 DATA BIN 11111111
 500 REM ***********************
 503 REM  INF. DER. FICHA BLANCA
 505 REM ***********************
 510 DATA BIN 00000110
 520 DATA BIN 00000110
 530 DATA BIN 00001100
 540 DATA BIN 00001100
 550 DATA BIN 00011000
 560 DATA BIN 01110000
 570 DATA BIN 11000000
 580 DATA BIN 11111111
```

```
600 REM ***********************
603 REM  SUP. DER. FICHA NEGRA
605 REM ***********************
610 DATA BIN 00000000
620 DATA BIN 11000000
630 DATA BIN 11110000
640 DATA BIN 11111000
650 DATA BIN 11111100
660 DATA BIN 11111100
670 DATA BIN 11111110
680 DATA BIN 11111110
700 REM ***********************
703 REM  SUP. IZQ. FICHA NEGRA
705 REM ***********************
710 DATA BIN 10000000
720 DATA BIN 10000011
730 DATA BIN 10001111
740 DATA BIN 10011111
750 DATA BIN 10111111
760 DATA BIN 10111111
770 DATA BIN 11111111
780 DATA BIN 11111111
800 REM ***********************
803 REM  INF. IZQ. FICHA NEGRA
805 REM ***********************
810 DATA BIN 11111111
820 DATA BIN 11111111
830 DATA BIN 10111111
840 DATA BIN 10111111
850 DATA BIN 10011111
860 DATA BIN 10001111
870 DATA BIN 10000011
880 DATA BIN 11111111
900 REM ***********************
903 REM  INF. DER. FICHA NEGRA
905 REM ***********************
910 DATA BIN 11111110
920 DATA BIN 11111110
930 DATA BIN 11111100
940 DATA BIN 11111100
950 DATA BIN 11111000
960 DATA BIN 11110000
970 DATA BIN 11000000
980 DATA BIN 11111111
1000 REM ***********************
1003 REM  LECTURA FICHA BLANCA
1005 REM ***********************
1010 FOR I=0 TO 7
1020 READ DATO
1030 POKE USR "A"+I,DATO
1040 NEXT I
1050 FOR I=0 TO 7
1060 READ DATO
1070 POKE USR "B"+I,DATO
1080 NEXT I
1090 FOR I=0 TO 7
1100 READ DATO
1110 POKE USR "C"+I,DATO
1120 NEXT I
1130 FOR I=0 TO 7
1140 READ DATO
1150 POKE USR "D"+I.DATO
1160 NEXT I
```

```
1200 REM ***********************
1203 REM  LECTURA FICHA NEGRA
1205 REM ***********************
1210 FOR I=0 TO 7
1220 READ DATO
1230 POKE USR "E"+I,DATO
1240 NEXT I
1250 FOR I=0 TO 7
1260 READ DATO
1270 POKE USR "F"+I,DATO
1280 NEXT I
1290 FOR I=0 TO 7
1300 READ DATO
1310 POKE USR "G"+I,DATO
1320 NEXT I
1330 FOR I=0 TO 7
1340 READ DATO
1350 POKE USR "H"+I,DATO
1360 NEXT I
1400 REM ***********************
1403 REM   SITUACION INICIAL
1405 REM ***********************
1410 PRINT AT 9,14;""
1420 PRINT AT 10,14;""
1430 PRINT AT 11,14;""
1440 PRINT AT 12,14;""
1450 STOP
1505 REM
1510 REM ***********************
1513 REM  RUTINA PARA CONVERSION
          DE COORDENADAS. SE SU-
          PONE QUE LAS COORDENA-
          DAS DEL PROGRAMA ORIGI
          NAL, SE ENCUENTRAN EN
          LAS VARIABLES X E Y.
1515 REM ***********************
1520 REM
1530 PRINT AT 2*X+1,2*Y+6;""
1540 PRINT AT 2*X+2,2*Y+6;""
```

## Distribución de los caracteres gráficos

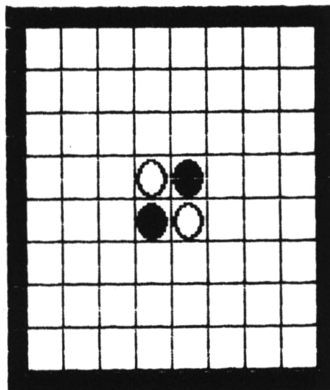
FICHA BLANCA:

A → Superior izquierda
B → Superior derecha
C → Inferior izquierda
D → Inferior derecha

FICHA NEGRA:

E → Superior derecha
F → Superior izquierda
G → Inferior izquierda
H → Inferior derecha

Para adaptar el programa al juego del Othelo dispone de una subrutina que se encuentra en la línea 1510 que habrá de llamarse cada vez que se quiera dibujar una ficha en el tablero en la posición X,Y. Tendrá que hacerse otra igual para la ficha negra. Y por último no se podrá usar SCREEN$ en la búsqueda, pero esto no es problema ya que ya hemos visto cómo guardar un tablero en el ordenador con ayuda de una matriz de caracteres.
Una vez haya introducido el programa, ejecútelo, y el tablero le quedará así:



## CARATULAS Y GRAFICOS

Si usted quiere comercializar sus programas tendrá que presentarlos de una forma atractiva, con una buena presentación, y lo que es también muy importante, con una buena portada (aunque es un engorro a la hora de cargarlo, ¿no?). Para realizar las portadas disponemos de programas comerciales que en esencia son parecidos al que se presenta en la página 5 que mueve el cursor por la pantalla aunque incorporan muchas más cosas, como por ejemplo llenar de color una figura, ampliarla, reducirla, trasladarla, etc. Dado que incorporan la facilidad de grabar la pantalla, usted puede hacerse sus dibujos y luego grabarlos en cassette o microdrive para que se incorporen a su programa como carátula. Con estos programas se pueden realizar dibujos muy completos. ¿Qué le parece este cuyo título podría ser: «El mejor programa de Europa»?



Para incorporar la carátula a su programa es muy sencillo. Primero usted crea el dibujo usando los medios que sean y luego lo graba en cinta con un nombre, por ejemplo «dibujo» del siguiente modo:

```
SAVE "DIBUJO" SCREEN$
```

o si es para microdrive:

```
SAVE *"m";1;"dibujo" SCREEN$
```

Si su programa se llama «programa» deberá crear otro programa que haga lo siguiente:

```
10 LOAD "DIBUJO" SCREEN$
20 LOAD "PROGRAMA"
```

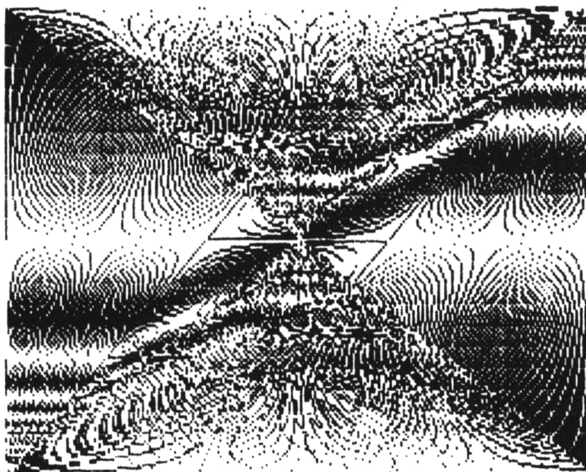
y grabarlo del siguiente modo:

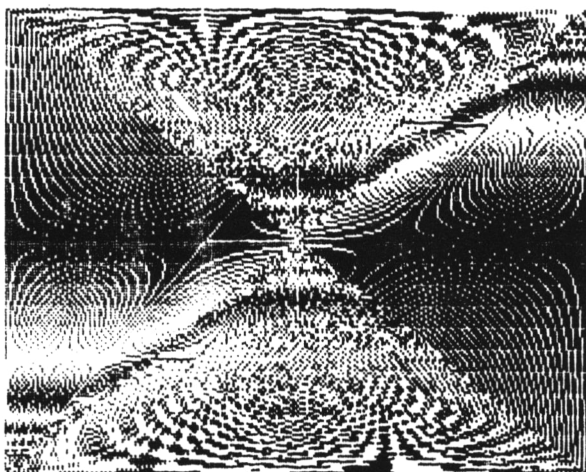
```
SAVE "cargador" LINE 10
```

El orden en la cinta debe ser: Primero «cargador», luego «dibujo», y por último programa. De este modo tenemos una carátula y un dibujo que nos distrae mientras «programa» se está cargando. El problema de esto es que perdemos alrededor de dos minutos y medio esperando que se cargue «dibujo».
Para solucionar esto podemos crear un programa que nos haga el dibujo y que sea más corto de cargar que el dibujo en sí. Por ejemplo este programa es muy efectivo y sólo tiene 18 líneas:

```
 10 FOR i=0 TO 255 STEP a
 20 PLOT 128,88
 30 DRAW OVER 1;-128+i,87
 40 NEXT i
 50 FOR i=0 TO 255 STEP a
 60 PLOT 0,0
 70 DRAW OVER 1;i,175
 80 NEXT i
 90 FOR i=0 TO 255 STEP a
100 PLOT 128,88
120 DRAW OVER 1;-128+i,-87
130 NEXT i
150 FOR i=0 TO 255 STEP a
160 PLOT 255,175
170 DRAW OVER 1;-255+i,-175
180 NEXT i
```

Cambiando los valores de a en las líneas 10,50,90 y 150 obtendremos cosas como éstas:





Para aquellos que se hayan entretenido en observar los programas de la cinta HORIZONS que viene con el Spectrum observarán que en todos los programas viene una rutina para modificar los caracteres, que nos permite hacer cosas como ésta:




Esta rutina se compone de dos partes, una en código máquina que se carga alrededor de la dirección 32000 y otra parte en BASIC que envía los parámetros a la rutina. Para usarla sólo tiene que grabar en su Spectrum la parte de BYTES del primer programa posicionando la cinta justo antes y escribiendo:

```
LOAD "" CODE
```

Luego observar el programa en BASIC que le precede. Y verá que hay parámetros para centrar el texto, tamaño de letra tanto horizontal como vertical (factor multiplicador), posición de comienzo y la frase en sí que se deposita en la variable D$. Luego hay una subrutina que le pasa los parámetros a la memoria para que la subrutina en código máquina escriba la frase. Es muy efectivo y rápido y soluciona el problema de los títulos.

**ACELARACION DE PROGRAMAS**

Para que los programas corran más rápidamente hay que seguir una serie de reglas. Estas reglas acostumbran a ser contrarias a la buena estructuración del programa, pero no estará de más disponer de las dos versiones.

Lo primero que debe hacer es suprimir todas las subrutinas e incluirlas en el sitio en que son llamadas. En caso de que tenga que poner alguna, bien porque no dispone de suficiente memoria, bien porque es llamada en muchas partes del programa, colóquelas siempre al principio, pues es por el principio por donde el ordenador empieza a buscar el número de línea de la rutina a la que se ha llamado.
Otra cosa que enlentece a los programas son las sentencias REM, deberá suprimirlas todas, o al menos procurar que no se ejecuten. Aunque esto último es un mal menor ya que cuanto más largo es el programa (incluidas las sentencias REM ejecutables o no), más tarda en encontrar una línea.
Por lo dicho en el párrafo anterior, también será importante, compactar al máximo el programa, es decir, colocar el máximo de sentencias posibles con el mismo número de líneas. Si hace esto tenga cuidado con las sentencias IF...THEN, ya que si no se cumple la condición, no se ejecutará nada de lo que haya en la línea.
En cuanto a las variables, hay que usar las menos posibles, procurando utilizar las que se hayan creado si no hace falta que continúen guardando su valor. También es interesante, pero en mucha menor medida, que los nombres de las variables tengan el mínimo número de caracteres. Tenga presente que el trabajo con matrices es mucho más lento para el ordenador que el trabajo con variables normales.
Otra cosa interesante con respecto a las variables es el hecho de inicializarlas (darle valor por primera vez) en un orden determinado: primero las que se utilicen más veces en el programa, y luego las que se utilicen menos veces, ya que ocurre algo similar a los GOTOs y GOSUBs, y es que cuando hay que usar una variable se empieza a buscar por las primeras.
Por último procure utilizar lo menos posible las sentencias IF intentando sustituirlas por operadores lógicos, tal como se hizo en la rutina de puntuación del juego del 15.

## USO DE LOS MANDOS DE JUEGO (JOYSTICK)

Para cambiar un programa que hace las entradas por teclado y adaptarlo al joystick lo primero que tiene que saber es qué puerto de entrada usa el mando que usted tiene. Si es un joystick de cursor y su programa usaba las flechas para mover objetos, entonces no tendrá que hacer ningún tipo de adaptación. En caso contrario, deberá cambiar las instrucciones de entrada. Normalmente estas instrucciones son del tipo:

```
IF INKEY$=...THEN...
```

Y usted deberá cambiarlas por instrucciones como:

```
IF IN xxxxx=...THEN...
```

Donde xxxxx se refiere al puerto de entrada para el que está programado su interface.

# INDICE

Pág.

PROLOGO .......... 7

ASPECTOS DE LA PROGRAMACION EN BASIC .......... 9
Los contadores. Generador de números aleatorios. Localización y movimiento de objetos en pantalla. Conjuntos.

JUEGOS INTELIGENTES .......... 17
Conocimiento del juego. Diseño de la estructura. Organigrama. Afinación o diseño arriba-abajo. Escritura del programa.

UN EJEMPLO SENCILLO: NIM .......... 21
Descripción del juego. Diseño del gráfico. Afinación. Comentarios sobre el programa.

EL OTHELLO: UN JUEGO DE VERDAD .......... 26
Descripción del juego. Iniciación. Para decidir quién empieza. Jugada del jugador. ¿Ha terminado la partida? Jugada del ordenador. Programa principal. Subrutinas.

COMENTARIOS SOBRE EL PROGRAMA DEL OTHELLO .......... 41

EXAMINADO EN PROFUNDIDAD: BUSQUEDA POR NIVELES 45

EL OTHELLO A NIVEL 2 .......... 53
Comentarios sobre el programa.

JUEGOS CON TRUCO: EL 15 .......... 63
Variables utilizadas.

JUEGOS GRAFICOS .......... 73
Variables usadas

GENERALIDADES .......... 81
Presentación de los programas. Carátulas y gráficos. Aceleración de programas. Uso de los mandos de juego (Joystick).

# OTROS TÍTULOS PARA EL SPECTRUM DE EDITORIAL NORAY, S.A.

**ZX SPECTRUM - QUÉ ES, PARA QUÉ SIRVE Y CÓMO SE USA**
por Tim Langdell

Este manual es el libro indispensable para todo aquél que quiera conocer el fantástico mundo de este ordenador. Empieza en cómo conectarlo y acaba dejando al lector en un grado más que elevado para llevar el Spectrum al máximo. Adaptado al Spectrum + y Spectrum normal.

**PROFUNDIZANDO EN EL ZX SPECTRUM**
por Dilwyn Jones

Para los que no se conformen con los manuales, este libro profundiza en los secretos del ZX Spectrum. Tanto si quiere profundizar en el ROM, como si quiere divertirse con un juego en tres dimensiones, en este título encontrará toda la información necesaria.

**ZX SPECTRUM, APLICACIONES PARA LA CASA Y LOS PEQUEÑOS NEGOCIOS**
por Chris Callender

El ZX Spectrum es un ordenador que no sólo sirve para juegos. En esta obra se explican quince programas prácticos para el hogar y el negocio. Directorios, contabilidad, gráficas, stocks, calendario, etc.

SOFTWARE: CINTA CASSETTE INCLUYENDO LOS 15 PROGRAMAS QUE SE TRATAN EN EL LIBRO «ZX SPECTRUM, APLICACIONES PRÁCTICAS PARA LA CASA Y LOS PEQUEÑOS NEGOCIOS».

## CÓMO CREAR TUS PROPIOS JUEGOS PARA EL ZX SPECTRUM
por Ramón Rovira (en preparación)

Este libro está concebido para que cada uno pueda diseñarse sus propios juegos y no tenga que conformarse en copiar y jugar con los ya clásicos y repetitivos. De forma clara y con ejemplos prácticos enseña a diseñar juegos inteligentes, de aventuras, marcianos, etc.

## 18 JUEGOS DINÁMICOS PARA TU DRAGON
por Pierre Montsaut

En esta obra se presenta una colección de 18 juegos programados en Basic. En los diferentes juegos se utilizan las funciones propias del microordenador, sonido, color, gráficos de alta resolución, etc. Después de una introducción al juego, se explican posibles modificaciones para adaptarlo al estilo de cada jugador.

## DICCIONARIO MICROINFORMÁTICO
por Ramón Tapias

Esta obra es un elemento indispensable para todo aquél que se lanza al mundo de la informática y debe aclarar conceptos o que debido a la falta de bibliografía en nuestra lengua se ve obligado a consultar obras en inglés.
De forma clara y razonada, va analizando uno a uno los principales vocablos de este fascinante mundo.
Contiene un vocabulario Inglés/Español.

La mayoría de los usuarios de microordenadores dedican la mayor parte del tiempo al ocio que aportan los juegos. Muchas veces los juegos que se encuentran en el mercado no se adaptan a los gustos o necesidades propias de cada usuario, o bien llegan a hacerse monótonos cuando se han practicado varias veces.

Pensando en todo esto, el autor ha concebido este libro para que cada uno pueda crear sus propios juegos o bien adaptar los ya existentes.

Jugar y aprender podría ser el subtítulo de la obra ya que en el libro se repasa la teoría de la programación y los trucos más adecuados para los juegos.

En este libro el lector encontrará la explicación de cómo hacerse sus juegos desde el más sencillo (NIM), pasando por complicados juegos inteligentes para acabar con los clásicos juegos gráficos de fantasmas o marcianos.

EDITORIAL NORAY, S.A.
San Gervasio de Cassolas, 79
Barcelona